Speaker Identification and Verification Applications

Internal Working Draft -- February, 2006

VoiceXML Forum
Speaker Biometrics Committee

About the VoiceXML Forum

Voice Extensible Markup Language (VoiceXML) is a markup language for creating voice user interfaces that use automatic speech recognition (ASR) and text-to-speech synthesis (TTS). Since its founding in March 1999, the VoiceXML Forum has continued to develop, promote and to accelerate the adoption of VoiceXML-based technologies via more than 150 member organizations worldwide.

Tens of thousands of commercial VoiceXML-based speech applications have been deployed across a diverse set of industries, including financial services, government, insurance, retail, telecommunications, transportation, travel and hospitality. Millions of calls are answered by VoiceXML applications every day.

The Forum's primary focus areas include:

- Promoting the adoption of VoiceXML-based technologies
- Cultivating a global VoiceXML ecosystem
- Actively supporting standards bodies and industry consortia, such as the W3C and IETF, as they work on VoiceXML and related standards, such as CCXML, X+V, MRCP, and speech biometrics.

For more information on the VoiceXML Forum visit the website at **http://www.voicexml.org**.

Disclaimers

Revision History

Date	Description
February, 2006	Internal Working draft

Content

1. Goals of the SIV-Applications Document

The main goal of this document is to provide a good representative set of sample SIV applications from a technical perspective. It should complement the SIV-requirements document by the same working group and serve as a test-set for the development of an SIV-extension for VoiceXML. The set of sample applications should cover the most important markets/industries and the most important application categories.

The document has the following main parts:

- Overview about markets and existing applications.
- Future applications based on voice authentication.
- Implementation of above applications based on an ad hoc VoiceXML-extension for SIV.

The examples might be used to check the validity and compare different layouts of the SIV-extension. Therefore the application descriptions should be as realistic and detailed as possible with respect to their SIV-processes. An application description should contain at least the parts designation, authentication (verification or identification) and enrollment. The authentication part usually includes the handling of a rejected speaker. The enrollment part includes the evolution of the voice-model through adaptation.
For each application the benefits of using SIV in this particular case are listed.
For existing ("real world") applications we state the name of the application or product and the owner or implementer of it and we also credit the vendor of the verifier underlying the SIV-part of the application.
In a classification section we assign the attributes industry, application category (see next chapter), scale of deployment and security level to each application.
In a "technical features" section the authors highlight the special features of each application and link it to the other documents produced by the same working group.

This document (at least at the current stage) includes also applications that might not be very likely to become implemented in VoiceXML. The focus will, however, remain on VoiceXML applications and applications that could conceivably be implemented using voice XML. We state the estimated suitability of the application for being implemented in VoiceXML in its classification section.
The ranks range from low to high. Main indications for VoiceXML-relevance are

- SIV done over a phone.
- SIV embedded in an IVR dialog.
- SIV used with concurrent ASR, e.g. for challenge response.
- The application has been implemented using VoiceXML with a vendor-specific SIV-extension.

If two or more of these criteria are true for an application, we give it a high relevance, if only one criterion is true, the application gets a medium relevance and if none of the criteria is true for the application, it gets a low VoiceXML-relevance.
We encourage everybody to send in their own examples to this document. Examples with a high VoiceXML-relevance according to the above definition are preferred, but also applications that don't meet the above criteria can be included, in particular if they have particular technical features with respect to SIV, which are not covered in the examples so far.

Please send your examples to biosig@voicexml.org or to claudia@ibp.de and martin.eckert@t-systems.com. If you have questions concerning the submission of an example please contact claudia@ibp.de and martin.eckert@t-systems.com.

2. Classification of SIV-applications

Today we can find SIV-based applications in a lot of different industries (see the list below). The majority of these applications falls into one of the main application categories listed below. Not all of them are telephony-based. We find a variety of different architectures underlying the SIV-applications. The architecture components are

- Voice input device, e.g. telephone or terminal with microphone connected
- Application-Runtime
- SIV-Engine, where the actual verification process takes place
- Storage of voice models, e.g. database near verification server, database at application server (VoiceXML), smartcard etc.

There can be further components involved, e.g. telephony server (VoiceXML-platform), application server, resource server (MRCP) etc.
The different components of an SIV application can be interconnected through various networks. In the sample applications we find for example the classical VoiceXML-architecture (“mobile phone contract details” or “direct banking service”), standalone solutions (“PC access” or the automotive example “car theft protection”), or locally distributed (LAN) architectures as in “door lock access”. In the future applications section we also outline examples, where smartcards are used for storing voice enrollment data (“access control for ATM”).
For further details see [Best Practices].

Market Situation
To be updated

Industries

- telecom
- financial services
- call centers (outsourced)
- corrections industry
- media
- military
- government (forensics, law enforcement)
- value added services
- information technology
- consumer market
- health care
- transportation industries
- private industries
- large organizations
- general

Application categories
Main application types (or categories) are the following:

- Transaction authentication:
    - authentication for self-administration through IVR over phone
    - password reset over phone (special case of self-service)
    - call center authentication
    - prison telephone usage
- Personalization of IVR dialogue:
    - use of speaker dependent voice technology (e.g. emotions detection)
    - use of speaker dependent information
- Information Retrieval:
    - Customer Information for Call Center
        - general features like age or gender

        - identification of specific individuals
    - Speech Skimming
- Access control:
    - voice-opened locks for physical devices (door, car, ...)
    - access control for computers and data networks
    - border control
- Remote time and attendance logging:
    - Employee time tracking
    - Home parole verification
- Audio mining:
    - Automatic indexation of audio broadcast
    - Forensics on intercepted calls

3. Current SIV-Applications

This section contains real world applications. Most of them are recent deployments. The application descriptions are ordered by application category.

3.1. Examples for Transaction Authentication

3.1.1. Access to Mobile Phone Contract Details

This application allows people to manage their mobile phone contract details over the phone. For additional security (a phone might be stolen) the application uses voice authentication. This application is medium security critical. No large amounts of money can be transferred, but a thief could change the contract details to allow making expensive calls on the owners account. Three variants of this application where considered in the planning phase. Scenario A was realized as a pilot project using the Magellan VoiceXML-platform and SIV-extension. Scenario B was realized as a pilot using the Magellan C-API with SIV-extension. In both cases the underlying verifier and ASR where Nuance 8.0.

Scenario A: Access to Mobile Phone Contract Details (Single-User and Single-Device)

In this scenario each client gets one TAN for enrollment.
Designation: To get access the client must call in from their mobile phone and the calling number must be transmitted. If no valid mobile phone number is transmitted, the caller is rejected. The application looks up the TAN for the phone number in the TAN-database at the beginning of the call. If the TAN is marked as already used, the application proceeds with authentication, otherwise with TAN-entry for enrollment. The calling number is used as the voiceprint id.
Authentication: To make deceiving the verifier with a recording more difficult, the application uses randomly generated digit sequences as prompted phrases (challenge response). For better results, the caller must repeat each four digit sequence twice e.g. „4523 4523. For additional user friendliness it uses variable length verification. Depending on the verification score the caller has to repeat between one and three digit sequences. To rule out attempts to break in, the application maintains a counter of the failed attempts to speaker verification. After a certain number of failed attempts in a row (can be on one call or on multiple calls), the application is closed for this phone number and the user can only re-open it by ordering a new enrollment TAN.
Enrollment: For enrollment the caller has to repeat six fixed four digit sequences. Again the caller must repeat each four digit sequence twice. If a speaker gets to the enrollment dialog for the second time with a new TAN, the old reference model is overwritten by the new enrollment. The verifier can be set to automatically adapt the reference model whenever the speaker was verified. Then in most cases the number of utterances needed for verification will get down to only one utterance.

Technical Features

- unique identity claim (no identification)
- transmitted calling number (ANI/CLI) is the voiceprint id
- concurrent processing SIV+ASR
- text-prompted mode, rollback or prompted phrase required
- challenge response
- uses randomly generated digit sequences as prompted phrases
- SIV-session with multiple turns
- interaction turn <field>
- uses variable length verification
- handles SIV-specific events: invalid phrase, session-end: accepted, session-end: rejected
- uses generic verification properties: Prompted Phrase, MinUtterances, MaxUtterances
- optional automatic adaptation (without requiring changes in the dialog)
- enrollment with a fixed number of fixed phrases
- access to enrollment controlled by TAN

- the decision between enrollment or authentication comes from the TAN database, not the reference model database
- existing reference models are overwritten by enrollment
- limited number of failed verification attempts

Scenario B: Access to Mobile Phone Contract Details (Multiple-User and Single-Device)

This scenario extends the first one in that more than one person can maintain a mobile phone account. In this case speaker identification (multi verification) is used instead of (single-) speaker verification. For each account, a certain number (*x*) of TANs are handed out, so that up to *x* speakers can enroll for it.
Designation: The transmitted phone number is now the key for the group of voiceprints. At the beginning, the application checks with the database if no voiceprint exists for this account. If so, it tries to fetch the corresponding TANs and proceeds with TAN-entry for the enrollment dialog. Otherwise it starts with verification.
Authentication: The authentication dialog is the same as in Scenario A, but the underlying authentication process now is small-group identification (multi-verification). Furthermore the authentication utterances are buffered for potential further use in enrollment or adaptation. As in scenario A, the account is closed, if authentication fails for three times in a row. In this case all users have to re-enroll with new TANs.
Enrollment and adaptation: For the first speaker to enroll, the behavior is identical to the behavior in example one. When a second speaker with a valid TAN wants to enroll, he will first go through identification which will fail. Then he will be asked, if he wants to enroll and be transferred to the TAN-entry dialog. The utterances already made by the caller in the identification dialog are used in addition to the usual enrollment utterances to create the new reference model. Automatic adaptation is not recommended with multi verification, because if a speaker gets a positive result on more than one of the reference models in the group, then all those reference models would be adapted, spoiling the ones that don't belong to the speaker. Instead the application does a manual adaptation of the highest ranking reference model in the group, and it does so only when more than one utterance was spoken.

New features with respect to scenario A:

- speaker identification with calling number as group key
- manual adaptation
- buffering of verification utterances
- buffered utterances are used for enrollment in addition to usual enrollment utterances

Scenario C: Access To Mobile Phone Contract Details (Multiple-User and Multiple-Device)

This scenario extends the second scenario in that the application can be used also from another phone, for example from the conventional telephone network.
Designation: If the application is called with a valid mobile phone number as calling number, the application assumes that this number also designates the account to be maintained and proceeds as scenario B. If no valid mobile phone number is transmitted, the caller is asked for his mobile phone number. This first utterance is buffered and might be used for the following verification. When the caller has said and confirmed the number, the application checks the database for enrollment TANs as in the previous scenario and proceeds with identification, if a voiceprint has been enrolled for this account. If the claimed phone number is valid, but no voiceprints have been enrolled for it, the caller is told, that enrollment is only possible from the corresponding mobile phone. The dialog prompts with an appropriate error message if the caller said an invalid phone number.
Authentication: If a buffered utterance exists from the designation phase, this is processed by the verifier first. Otherwise the authentication dialog remains the same as in scenario B. To keep up protection against recordings, the caller must repeat at least one random digit sequence with a successful verification result, even if the buffered utterance already leads to a positive verification result. Thus when the call is not made on the mobile phone whose account is to be processed,

the variable length verification takes from 2 to 4 utterances instead of from 1 to 3 utterances. Furthermore, if the cumulative decision is „accepted“ after the second utterance the incremental result on the second utterance must also be positive.
Enrollment: Enrollment and adaptation work as in scenario B. The application has no separate reference models for the different channels. To prevent verification errors resulting from the different possible input channels at least in the main case, where the client calls in from his mobile phone, the application ensures that enrollment is always done from the original mobile phone. Note however that the reference model will always be adapted, if a speaker is identified calling in from another phone. This is because in this case there will always be at least two verification utterances including the buffered designation utterance.

New features of this example with respect to scenario B:

- it has a „designation phase“, an utterance is buffered before the voiceprint id is known
- variable length verification with the minimum and maximum number of utterances determined at runtime
- cumulative results and incremental results used
- users can call in on different channels

Classification:
- industry: telecom
- type: authentication for self administration
- security level: medium
- scale of deployment: national
- VoiceXML-Relevance: high
- VoiceXML-Sample-Code: 4.1-4.3

Benefits:
- increased security of the service
- increased flexibility (Scenario C)

3.1.2. Authentication for Direct Banking Service

This real world example was contributed by Judith Markowitz. The original source is:
First Direct Bank VoiceID Quarterly Volume 6, Issue 2 April, 2002 p1, 5.

Customers of „First Direct Bank of Israel Leumi“ are authenticated through background speaker verification during live conversations with a call-center agent. The application contains an alternative knowledge based authentication option. Callers are not indicated at calling time that speaker verification or enrollment are taking place, but all customers of the bank are informed in writing of this background authentication procedure. The underlying verifier is FreeSpeech™ from Persay.

Designation: The caller is asked by an IVR-System to enter his unique ID through DTMF-tones. He is then transferred to the call-center agent who will handle his actual banking requests. The callers name and account numbers as well as the status of his voiceprint are displayed on the agents screen.
Authentication: When a voiceprint exists for the caller, then background speaker verification is performed during the caller’s conversation with the call-center agent. Whenever authentication is needed during the call, the FreeSpeech system is asked for the current score and decision. If the verifier has not reached accepted the speaker, either because of insufficient data (only the agent has spoken), or if the speaker is rejected by the verifier, then the knowledge-based authentication alternative is used.
Authentication alternative (knowledge-based): The call-center agent is guided (graphically on his workstation's screen) to ask verification questions. The correct answers are stored in the very

first call of the customer to the service using a TAN. If the caller can't answer the questions he'll get a new TAN.
Enrollment and adaptation: The system automatically checks if the caller has completed enrollment. If not, the authentication alternative is used. If the secret questions were answered correctly, the call will be sent to enrollment. The system limits the maximum length of a single enrollment segment, though. Usually it takes three subsequent calls to the banking service to collect sufficient speech data for creating the voice model. The system automatically creates the reference model when there is sufficient audio.
Reference models are not automatically adapted in this application. There is a background process of reject analysis performed by the system administrator. This process uses the system reports to expose the high "reject rate" callers and to correct their voiceprints.

Classification:

- industry: banking
- type: call center authentication
- security level: very high
- scale of deployment: national
- VoiceXML-Relevance: medium
- Closest VoiceXML-Sample: 4.6, but with verification instead of identification

Benefits:

- increased security compared to PIN-Code-Only-Authentication.
- increased user friendliness compared to Two-Factor-Authentication with mandatory verification question.
- reduced call length compared to Two-Factor-Authentication with mandatory verification question.

Technical Features

- enrollment and verification on live free speech
- interaction turn is bridged transfer
- detection of recordings through human operator
- access to enrollment controlled through knowledge-based question
- enrollment distributed on several calls
- authentication by SIV with fallback on knowledge
- system is language independent

3.1.3. Password Reset

This service is used for password reset on multiple password-secured systems inside a company or organization. Each employee obtains a unique id and an enrollment TAN for the password reset service. The unique id is the voiceprint id; the employees name is used for enrollment and for verification. When a user has forgotten one of his passwords he calls into the password reset service. If the user is successfully authenticated he can name one or more applications for which he wants his password reset, e.g. „Windows". After checking the users access rights to the specified applications, the system acknowledges, if the password reset was successful.

Designation: The service first asks for the employees' unique id. If a reference model exists for this id the service proceeds with authentication. Otherwise with enrollment.
Authentication: The caller is asked for his name, which will be verified.
Enrollment: The caller is asked to enter his TAN via DTMF-tones. If the correct TAN was entered he is asked three times to say his name. If the utterances where consistent, the reference model is generated from the three utterances.

Classification:

- industry: private industries, medium to large organizations
- type: password reset
- security level: medium
- scale of deployment: company-wide
- VoiceXML-Relevance: high
- Closest VoiceXML-Sample:

Benefits of SIV compared to password reset initiated by human operator:

- increased security
- lower costs
- relief of the system administrator
- around the clock availability
- not embarrassing for users
- faster

Technical Features:

- voice enrollment on free speech (phoneme grammar)
- text-dependent verification

3.2. Examples for Access Control

3.2.1. SIV controlled door locks

This real world example was contributed by Judith Markowitz. The original source is:
The Manor at York Town VoiceID Quarterly Volume 6, Issue 4 3 October, 2002 p2-3

The apartments at „The Manor at York Town“ retirement community, can be opened by their residents and by other authorized persons through speaker identification. The voiceprints for the community are stored on a central server. Access rights are managed for each apartment separately. They can be time dependent, e.g. for housekeeping staff. Enrolled speakers get a smartcard containing a unique PIN, which is used in combination with speaker verification to gain access to perimeter doors and community facilities such as a self service shop that is open 24 hours and needs no cashiers, since the customers are identified by the systems and all their purchases automatically booked from their accounts. The smartcard is not needed for door access.
A secondary use of the SIV-controlled door-locks is of the remote time and attendance type: The administrator can monitor the status of residents by the frequency of their voice. If a resident has not use his or her voice to access a door in a given period of time, security can be notified to see if that resident is ill or is having a problem.
A third possible use is of the personalization type: The voices of individual members of housekeeping can be tagged with work orders so that when a housekeeper arrives and is authenticated by voice, a work order will appear on a monitor in the room.

SIV in this application is based on speaker authentication from Graphco Technologies.

Designation: Each apartment door has a special button for speaker authentication. When it is pressed, the apartments ID is transmitted to a central server which assembles the corresponding group of authorized persons with respect to time and date.
If a smartcard is entered at a perimeter door, the central server checks, if the owner of the smartcard has the right to access the specific perimeter door. If so, it performs single speaker verification against the corresponding voice model.

Authentication: is performed by speaking freely for about 1.5 seconds while keeping the door button pressed. If speaker authentication fails one can retry or use a key or key pad.
Enrollment: is done at a special office, not at the doors themselves, but with the same kind of microphone arrangement as the doors have. The system builds the reference model from about 3 minutes of free speech. An administrator observes the identity of the person enrolling and manages their access rights. If residents decide to give access to other persons, e.g. frequent visitors, they must arrange the enrollment with the administrator.

Classification:
- industry: apartment residence
- type: access control for physical device
- secondary type 1: remote time and attendance
- secondary type 2: personalization
- security level: medium
- scale of deployment: a restricted person subgroup (residents, visitors, house keepers, nursing stuff)
- VoiceXML-Relevance: low
- Closest VoiceXML-Sample: 4.4, but with verification instead of identification

Benefits:
- increased comfort
- increased security
- additional possibilities like automatic shopping

Technical Features:
- voice enrollment and verification on free speech (phoneme grammar)
- text-independent verification
- multi-verification on small group
- group of authorized users is not fixed. It is time dependant and managed through a central server.

3.2.2. Voice-Coded Car-Theft Protection

VOCAL SCW 1 is a voice-coded anti car theft device for motor vehicles. It recognizes up to 5 authorized drivers by voice and password. The product consists of 2 components: microphone-speaker-unit for speech input and output with dialogue function to be mounted anywhere within the driver's area of the cockpit and control unit with voice-identification processor and actuator for the switch signal. It can be retrofitted in any vehicle. The system disconnects 3 electrical circuits when armed, e.g. ignition, starter and fuel pump. The system arms itself about 8 minutes after turning off the ignition or immediately on opening the door when the ignition is turned off. Together with the product, the owner obtains five secret codes, which can be given to five different authorized persons. The PINs consist of five numerical digits each. Numerical codes are entered at the speech input device through a tone dialer, which comes with the product. The PINs can be used for enrollment and re-enrollment (change of voiceprint and password), as a fallback when voice authentication fails, to turn into garage mode (allow temporary access with a chosen four-digit access code), and to switch on and off prompting of the system.

Designation: No designation is needed for identification, since it is performed on all the stored voiceprints. Designation for enrollment is done by entering one of the five valid PINs through the dialer. The PIN is also the voiceprint identifier.
Authentication: When the driver switches on the ignition he is prompted by the system to say his enrolled password. If authentication fails, the speaker has three retries. Then he can switch on the ignition again and has another three tries, before the system is closed for speech input. The user then has to use a PIN-code entered through a tone dialer. He has five tries to enter the correct PIN code.

Enrollment and adaptation: To enroll the user must switch on the ignition and after the prompt enter his PIN followed by “2” (the code for training). Then he is asked to choose a code word which has between 3 and 4 syllables. Then he must speak the code word within 2 seconds. The utterance is replayed by the system, to allow the user to check that the volume and timing were correct. This is repeated 7 times in a row. The initial reference model is build from this data. The first five times the user authenticates with the system, a special adaptation process is performed. No other speakers can be enrolled during this phase. Afterwards authentication is faster, but still some adaptation is constantly performed. The system usually reaches it’s maximum security level after about another ten successful authentications. If constantly used the reference model should evolve with the users voice and there should be no need for re-enrollment.

Classification:
- industry: automobile
- type: access control for physical device
- security level: medium
- scale of deployment: small group
- VoiceXML-Relevance: low
- Closest VoiceXML-Sample: 4.4, but without the fingerprint verification

Benefits:
- increased theft protection

Technical Features:
- voice enrollment with speaker defined password
- enrollment with phoneme grammar
- enrollment is text-dependent (enrollment utterances must be consistent with respect to phonemes)
- speaker verification is text-dependent (phoneme sequence must match the one enrolled)
- implicit multi-factor authentication (voice and knowledge) achieved through text-dependent verification on the enrolled password
- identification on small group
- identification on all stored voiceprints (no designation for identification)
- PINs used as authentication alternative and key to (re-)enrollment
- PIN is voiceprint id, allows re-enrollment of individual speakers
- combination of SIV and DTMF, no ASR needed

3.2.3. Personal Computer Access Control

In the PC operating system Mac OS 9 speaker verification is used as an alternative login procedure besides the usual typed password. The speaker verification in this application is based on Nuance verifier.
Designation: At startup the user selects his login name from the list of registered users and clicks a login button. If no voiceprint exists for his account, he proceeds with the usual password entry via the keyboard, otherwise the user proceeds with the authentication alternative via SV.
Authentication alternative (SV): A window pops up, containing the request to say the pass phrase for speaker authentication and the textual representation of the pass phrase, as specified during enrollment (if not configured to be hidden). The window also contains a display, where the waveform of the speech is drawn while the user is speaking, thus allowing the user to verify that the microphone is working correctly. If the user fails three times to be verified by the system, a hint pops up, that maybe a new voiceprint should be recorded and the system proceeds with the textual password entry.
Enrollment: For enrollment, a user with the privileges to access user accounts must be logged in. In the user enrollment window, the textual representation of the pass phrase can be entered. It will be displayed at login time in the authentication window. Either the standard phrase “my voice is my password” can be used, or a different text entered. This doesn’t need to match the words of

the spoken phrase, it can also be a hint like “What is the name of your mothers high school?”, but the standard way is to use the exact wording to make sure that the pass phrase can’t be forgotten. For enrollment the user has to record the pass phrase for four times. As in the authentication step a graphical display of the wave form assists in making sure that recording works properly. The user also has the opportunity to play back each recording. When the user has completed the four recordings the system tries to build a reference model from them. If the utterances where inconsistent, a warning is displayed. If no inconsistencies where detected or the user decides to ignore the warning, the dialog proceeds with a test of the pass phrase. The user can try for three times to get verified with the pass phrase he enrolled. If he succeeds, enrollment is completed successfully, if not, the user must redo the four recordings.
This application has no adaptation procedure. Re-enrollment, which overwrites the previously stored reference model, is possible at any time through a user with administration rights. Re-enrollment is recommended explicitly by the system, if authentication fails repeatedly.

Benefits:
- pass phrase will not be forgotten, if display of textual representation is chosen
- users need not be capable of typing, e.g. pre-school children, or handicapped people
- increased security, because a more difficult written password can be used, e.g. a long random character sequence, since it will be needed only in the rare case, when voice authentication fails

Classification:
- industry: personal computer
- type: access control for computer
- security level: medium
- scale of deployment: small group
- VoiceXML-Relevance: low
- Closest VoiceXML-Sample: 4.4, but without the fingerprint verification

Technical Features:
- voice enrollment using standard pass phrase by default
- optional speaker defined pass phrase
- enrollment with phoneme grammar
- enrollment is text-dependent (enrollment utterances must be consistent with respect to phonemes)
- enrollment utterances are recorded and optionally played back to the enrollee
- speaker verification is text-dependent (phoneme sequence must match the one enrolled)
- speaker verification is text prompted by default
- implicit multi-factor authentication (voice and knowledge) can be achieved by hiding the textual representation or using only a hint
- password used as authentication alternative
- enrollment controlled by user administrator

3.2.4. PC-Access 2: Multi-Biometric authentication with USB-camera

Details to be updated. BioID uses face, voice and lip movement to identify a person. The user looks into a standard video camera and says his or her name to get authenticated.

The BioID software development kit (SDK) 3.0 can be implemented in any application where people must be identified. The SDK offers a user enrollment wizard, user recognition including identification and verification, user template and authorization management as well as template storage to a database, local PC, or smartcard, and enrollment management.

Other features include an adaptable client interface, support for BioAPI, including the BAPI and H-API specifications, and an improved algorithm for increased recognition speed and accuracy for an unlimited number of users. An improved graphical user interface (GUI) features an animated software agent to assist users in setup, enrollment, and classification. The BioID template size has also been reduced to 16 KB in the latest version of the SDK, so templates may be easily transferred over the Web for Internet-based applications.

Classification:
- industry: personal computer
- type: access control for computer
- security level: medium
- scale of deployment: small group
- VoiceXML-Relevance: low
- Closest VoiceXML-Sample: 4.4, but without ASR, multiple verification modes simultaneously active

Benefits:
- increased comfort
- increased security

Technical Features:
- multi-biometric verification and enrollment
- voice enrollment on speaker-defined pass-phrase
- enrollment with free speech (phoneme grammar)
- implicit multi-factor authentication (biometric and knowledge) achieved through text-dependent verification on the enrolled pass-phrase
- multi-verification on small group

3.2.5. Border Control

The US PORTPASS Program, deployed at remote locations along the U.S.–Canadian border, recognizes voices of enrolled local residents speaking into a handset. This system enables enrollees to cross the border when the port is unstaffed. *Details todo.*

Classification:
- industry: government
- type: border control
- security level: high
- scale of deployment: small group
- VoiceXML-Relevance: low
- Closest VoiceXML-Sample: 4.4, but without ASR, no fingerprint

Benefits:
- increased comfort
- increased security

Technical Features: as in car-theft-protection

3.2.6. Airborne Interactive Response System (AIRS)

The speaker verification system represents an access control constituent operation of the CPU of an integrated system aboard an airborne transport vehicle. Functional aspects include initiation of log on sequence; voice template indexing and retrieval, parallel system communication link;

analysis of verbal response to authentication process; concurrent scoring; authentication; and alarm parameters.

System Initiation: Personnel seeking access to controlled space initiate speaker verification process by verbal command “LOG ON”. A resulting system generated prompt of, “STATE NAME” results in text-dependent enunciation of name and position title by the individual seeking access.

Primary Authentication (SV): The corresponding value of the verbal response must be equal to stored voice templates for that employee. Retrieval of voice templates is conducted in this manner and a communications-link is established between primary and secondary biometric systems; ostensibly aircraft and ground-based locations. A numeric index value corresponding to identified voice template(s) is transmitted from primary to secondary biometric system, thus permitting both computers to identify the correct template(s) and concurrently analyze and scoring of voice responses for secondary authentication.

Secondary Authentication (SV): Upon establishing communication link between primary and secondary voice biometric systems, the primary system prompts employee to enunciate a series of system-selected words and numbers (text-prompted). Vocal responses permit primary and secondary biometric systems to analyze and score said vocal input. Although both systems concurrently analyze elements of the personnel’s speech, scoring by each system is autonomous. Both systems must concur on scoring of vocal input (within parameters) in order to authenticate the personnel seeking access to the controlled space. If either system fails to authenticate or is compromised, scoring would not be sufficient to authorize access to the controlled space.

If secondary authentication is inconclusive, such as might be caused by abnormalities in mode of communication, then an additional system prompt is given for personnel to recite different series of words and numbers selected by the primary system. Concurrent scoring is conducted in the same manner noted above.

If multiple attempts fail to authenticate personnel, or if both primary and secondary systems fail to authenticate, then an alert is transmitted to a designated location for verification or action.

Acoustic Monitoring (SV): Upon above-referenced system authentication, voice templates of authorized personnel are indexed relative to aircraft and duration of flight. No changes or alteration of templates is possible. Authentication, likewise, initiates acoustic monitoring sub-system to assure integrity of cockpit or controlled space is maintained for the duration of flight.

Far-field audio sensors detect all utterances or sounds within confines of the controlled space, and match audio input against indexed templates of authorized personnel. Comparison is conducted by the primary biometric system on a text-independent basis. Results producing a match between far-field audio input and the indexed templates produce no action. If comparison determines sufficient difference between far-field audio and indexed templates (non-match), a coded alert is transmitted to one or more ground-based locations for action.

Ingress or egress by authorized personnel results in system prompt “STATE NAME”, and personnel provide the text-dependent response, in accordance with “System Initiation”, and “Primary Authentication (SV)”

If response or verbalization by any individual indicate variance or shift in vocal frequency commensurate with condition of severe emotional tension or duress, then an alarm is transmitted to ground-based facility for review or action.

Enrollment: Process of enrollment utilizes the same or similar audio equipment including close and far-field microphones equivalent to those utilized for authentication and monitoring. Authenticity of personnel submitting to enrollment process is positively confirmed prior to recording individual voice templates. Text dependent phrase particular to each employee is established and personnel record the corresponding initial voice template. Next, personnel provide verbal samples comprising a varied vocal palette of constituent elements of speech, sufficient for text independent speaker verification to be utilized by concurrent secondary authentication and acoustic monitoring.

Classification:
- Industry: aviation
- Type: aircraft access control
- Security level: high
- Scale of deployment: large group
- VoiceXML-Relevance: high

Benefits:
- Vocal command "LOG ON" initiates system (ASR)
- Text-dependent response enables voice template retrieval (ASR, SV)
- Concurrent analysis by two biometric systems, increase accuracy and reduce risk of single system compromise or spoofing
- System-randomization of pass-phrase (text-prompted) reduces risk of spoofing
- Acoustic monitoring enables passive biometric means of assuring integrity of controlled-space
- Acoustic monitoring provides means of detecting breach of controlled-space
- Speaker verification and monitoring permit transmission of alerts, if parameters are exceeded.
- Detects vocal frequency shift, commensurate with emotional stress or duress

Technical Features:
- Voice enrollment (primary authentication) uses standard format
- Enrollment with phoneme grammar
- System delineation of acoustic speech from background noise, aircraft radio or synthesized voice alerts (TCAS, E-GPWS, etc.)
- Primary authentication is text-dependent (phoneme sequence must match one enrolled)
- Secondary authentication and acoustic monitoring is text-independent
- Concurrent SV systems, match/match=approved; match/no-match=denied; no-match/no-match=denied, alarm
- Additional system prompt of personnel to repeat new selected pass phrase if initial concurrent SV systems result in match/no-match

3.3. Forensics and Audio Mining

3.3.1. SIV on intercepted calls

The technology is offered for example by Persay. *Details to do*.

Classification:
- industry: government (police)
- type: forensics
- security level: high
- scale of deployment: internal use, speakers nationwide, possibly worldwide
- VoiceXML-Relevance: low
- Closest VoiceXML-Sample: *todo*

Speaker identification on intercepted calls

The application takes a list of intercepted calls and performs speaker identification on them.

Features:

- identification on recorded free speech
- identification on a group of voiceprints
- use of ranking list of possible speakers

Voice mining on intercepted calls

Find a particular speaker on a large series of recordings of intercepted calls. The recordings are sorted by verification scores for the specified speaker

Features:

- verification on recorded free speech
- use of numerical scores

3.3.2. BBNN Audio Mining

Details to do.

3.4. Further examples

Transaction Authentication for Self-Service (VoiceXML-Relevance: high)

- Telephone credit card purchases
- Telephone brokerage (Charles Schwab)
- The Home Shopping Network
- Bradesco Caller Auth for Banking (Nuance)

Transaction Authentication PIN reset (VoiceXML-Relevance: medium or high)

- Swisscom: Secure Employee PIN reset (Nuance)
- The Hartford: Secure Employee PIN reset (Nuance)

Access control to databases (VoiceXML-Relevance: high)

- TI Corporate Facility Access Control
- GE Polymerland Secure access to product information
- Gradiente: Secure access to personalized information
- Social Security Secure Access to online employee data (Nuance)
- Bell Canada: Secure field service automation (Nuance)

Access control for telephone lines (VoiceXML-Relevance: high)

- Toll Fraud detection (Sprints Voice FONCARD)

Law Enforcement (VoiceXML-Relevance: high)

- Home Incarceration (ITT Industries)
- UK Offenders: Tracking of youth offenders (Nuance)
- Prison Call Monitoring (T-Netix)

Other (VoiceXML-Relevance: high)

- Union Pacific (ScanSoft) (Information: http://www.speechtechmag.com/issues/7_6/voiceideas/1468-1.html)
- Mitel Secure access to Speak@Ease Auto Attendant (Information: see link above)

4. Future applications based on voice authentication

Future applications might involve technologies that are not yet on the market (as far as authors know):

- combination of voice and other biometric in a special phone or other input device
- storage of voice models on smartcards in vendor independent format with widely distributed card reading devices (e.g. ATM)
- Combination with speaker dependant voice processing devices e.g. dictation systems, emotions detector, ...
- Large group speaker identification without identity claim

Some conceivable future applications

4.1. Access Control

4.1.1. Voice-Only-Access Control for ATM

In this fictitious application, the ATMs of one or more banks read customers voice models from smartcards and authenticate customers through speaker verification against the model on their card.
Designation: The customer identifies for enrollment by inserting his conventional banking card into one of the banks enrollment terminals and entering his banking PIN. Since the reference model is taken from the smartcard, no further designation is needed for authentication,.
Authentication: The ATM asks the customer to repeat a randomly chosen sentence from a book (e.g. Grimm Fairy Tales). The sentence is displayed and also read out by the machine, but the customer can interrupt the machines reading.
Enrollment: Enrollment takes place at special terminals, which have two slots, one for reading conventional banking cards and one for writing voice models to the voice banking cards. For enrollment the customer has to read a whole paragraph from the book, sentence by sentence.

Note: This application assumes a very low false acceptance rate. The randomly chosen prompted phrases serve to prevent fraud based on recordings.

Classification:

- industry: banking
- type: access control
- security level: very high
- scale of deployment: national
- VoiceXML-Relevance: high (if implemented on phone)
- Closest VoiceXML-Sample: 4.1, but with phoneme or dictation grammar and required phrase in text format, URL for voice-model repository points to smartcard reader

Benefits:

- increased security
- no PIN-codes

Technical Features:

- text prompted verification and enrollment on phrases chosen randomly from a book
- voice model stored on smartcard

4.1.2. Multi-Modal Access Control for ATM with vendor independent biometric data

In this application the banking smartcard contains the customers fingerprint and some voice samples in a vendor independent format. Thus different banks can implement their ATMs with fingerprint and voice authentication of their choice.

Designation: As in the previous example a customer identifies for enrollment by inserting his conventional banking card into one of his banks enrollment terminals and entering his banking PIN. Since the reference model is taken from the smartcard, no further designation is needed for authentication.

Authentication: The ATM chooses from the data on the smartcard only the enrollment utterances suitable for the verifier used by the bank. It builds the reference model either on the ATM or transfers it to a remote verification server. The ATM asks the customer for authentication utterances and for fingerprint simultaneously. The authentication utterances depend on the implementation of the ATM. It might ask for random digit sequences or for the customers secret phrase or for a fixed phrase “my voice is my password”. After the transaction is completed, the voice model is deleted from the terminal or server.

Enrollment: For enrollment the customer has to repeat some digit sequences, that contain each digit at least once, he has to say a secret phrase and record a hinting question for it and he also repeats the fixed phrase “my voice is my password”.

Note: Since the smartcard contains audio data, instead of a voice model, encryption is particularly important in this example. (See also [Best Practices])

Note: In the application not only the verification technology but also the architectural details, e.g. where verification takes place, are implementation dependant.

Classification:
- industry: banking
- type: access control
- security level: very high
- scale of deployment: national
- VoiceXML-Relevance: high (if implemented on phone)
- Closest VoiceXML-Sample: 4.4

Benefits:
- increased security
- no PIN-codes

Technical Features:
- multi-modal verification and enrollment for voice and fingerprint
- smartcard contains a variety of encrypted enrollment utterances in vendor independent format

4.1.3. PIN-less Call-Center authentication

This fictitious example is similar to real world example 3.1.2 "Authentication for Direct Banking Service", but large group speaker identification is used to eliminate the PIN-entry-part of the dialog.

Designation: In this application designation is needed only for enrollment. This is done by transferring to the PIN-entry dialog and asking knowledge based questions. No explicit identity claim is asked from the caller for authentication, but the system uses transmitted calling numbers, if available, and recognizes implicit identity claims through name and hot word spotting for disambiguation (see below).

Authentication: Authentication is done on the fly while the customer is talking to an operator. The outcome of the identification is displayed to the operator when enough data is processed. It is possible that the identifier finds more than one “accepted” speaker (depending on thresholds). Instead of just returning the highest scoring speaker, the application tries to disambiguate by

making use of other available information, namely the transmitted CLI and verbal identity claims that it can spot in the callers speech. If a CLI is transmitted, that belongs to one of the registered customers, then name spotting is started simultaneously with identification with the name(s) belonging to the CLI. If the CLI and the name stated agree with the identification result, then it is very likely, that the system identified the right speaker. If the name spotting had no result, but the CLI coincides with the identification result, then there's also a high confidence in the result, but the operator might still confirm the name explicitly. If identification returns a result and the transmitted CLI does not correspond to any of the identified speakers, then the names of all the accepted speakers are loaded to the name spotting process, and the speech made so far is processed by the recognizer again with these names. (It has to be recorded for this secondary processing).
If disambiguation fails, the operator gets the list of accepted speakers on his screen, ordered by verification score, and has to disambiguate e.g. by asking for the callers name.
If the caller is not identified as any of the enrolled speakers or claims an identity, that is enrolled, but not among the list of accepted speakers, then the alternative authentication procedure by PIN and knowledge based questions is used.

Enrollment: This works similar to the enrollment in example 3.1.2. In addition the application must now store the callers (most used) CLI(s), if available, the callers name in a format that can be used for name spotting and the language the user speaks.

Note: This example may seem a bit far fetched, since today there is no identifier on the market, that can perform authentication on the fly on a group of speakers as large as the group of customers of a bank. This example was made up explicitly to expose some technical challenges to the VoiceXML-SIV-extension, in particular with respect to the simultaneous processing of bridged transfer, recording, identification, and ASR, which have no concurrent endpointing.
An example which resembles this one, but seems technically much more practicable is example 4.3.1 ("Voice Mail Tagging").

Classification:
- industry: financial service
- type: call center
- security level: high
- scale of deployment: registered customers
- VoiceXML-Relevance: high
- Closest VoiceXML-Sample: 4.6

Benefits:
- fraud prevention
- increased user friendliness

Technical Features:
- identification on live free speech
- access to enrollment controlled by human operator
- enrollment on free speech that was buffered during bridged transfer to human operator
- identification on all voiceprints in the database
- crosscheck with human operator
- identification not perceived by the caller
- simultaneous recording, ASR and identification during bridged transfer
- endpoint of recording connected to return of identification results
- identification continues when results are returned
- endpoint of ASR (name spotting) not connected to endpoint of verifier
- secondary ASR processing started in the middle of bridged transfer
- CLI and name spotting used for disambiguation of identification results

4.2. Personalization

4.2.1. E-Mail sending service

This application allows callers to send email-messages over a phone. A voice-message can be recorded and sent as an attachment to the e-mail. A special recognizer is used for spelling e-mail addresses, subject and sender of the message. Speaker identification is used to improve the recognition performance of the spelling recognizer by loading speaker dependant data.
Designation: The application first asks for the sender's name, which serves as a key to the identification group. Since different callers could have the same name, N-best recognition results are used to improve performance.
Authentication: The sender is identified by a combination of ASR scores and verification scores.
Enrollment: If the sender can not be identified by the authentication process, a new voiceprint as well as new speaker dependent data for the recognizer is stored. The designation utterance has been buffered and is used for the new reference model.

Classification:
- industry: telecom
- type: use of speaker dependant voice technology
- security level: medium
- scale of deployment: national
- VoiceXML-Relevance: high
- Closest VoiceXML-Sample: todo

Technical Features:
- large scale identity claim
- verification on name, which might not be in the grammar (rejected utterance buffered)
- name can be enrolled into a dynamic grammar
- same phrase for verification and enrollment
- no security issue
- speaker verification used to improve dialog by personalizing it

Sample dialog:
Users first call to the application:

C: Welcome to e-mail-sending-service. What is your name?
H: Claudia.
C: (Voice not known) I don't recognize your voice. Please repeat your name.
H: Claudia.
C: Now please spell your name.
H: C L A U D I A
C: Thank you. (Builds a voiceprint and a dynamic grammar entry from the first two utterances and a spelling model from second utterance.)
C: Now please spell your e-mail address.
H: C L A U 4 9 3 @ F O O . B A R
C: Thank you. (Stores the name and e-mail-address and builds a voiceprint and a dynamic grammar entry from the first two utterances and a spelling model from the last two utterance.)
C: Now please spell the recipient of your message.
H: S O M E B O D Y @ F O O 2 . B A R 2
C: Thank you. (Stores the recipient in users known recipients list)
C: < ask for further recipients>
C: Now please spell the subject of your message.
H: J U S T A R R I V E D
C: You may now record your message. You can stop the recording by pressing any key.

Secondary call to the application:

> C: Welcome to e-mail-sending-service. What is your name?
> H: Claudia.
> C: (Voice is known) Hi Claudia.
> C: (Loads my spelling model) Please spell the recipient of your message...

4.3. Information Retrieval Examples:

4.3.1. Voice-Mail Tagging

Messages spoken to the voice-mail (or answering machine) are processed by speaker identification against the list of known callers for the voice-mail-account. When the owner of the voice-mail-account expects an important message from someone in this phone book, he can initiate some special action to be taken, when a message is left by this caller, e.g. the voice-mail-service can send him a notification by e-mail.

Designation: If the calling number is transmitted, it is compared to the users phone book entries. It is used to disambiguate the identification result or to provide a default name for enrollment. The name for enrollment can be overwritten by the owner of the answering machine.

Authentication: Speaker identification and name spotting are performed simultaneously with all the enrolled voice prints and stored names. Also the transmitted CLI might be used for disambiguation.

Enrollment: Unknown callers can be enrolled (added to the list of known callers) by the owner of the voice-mail account by making use of their recorded messages. The owner can also initiate adaptation of voice models.

Classification:
- industry: telecom
- type: information retrieval
- security level: low
- scale of deployment: small group (personal phone book)
- VoiceXML-Relevance: high
- Closest VoiceXML-Sample:

Benefits:
- automatic indexation of messages
- detection of important messages

Technical Features:
- identification on recorded free speech
- access to enrollment controlled by human owner
- enrollment on free speech recorded messages
- no identity claim necessary
- identification on all voiceprints in the database
- transmitted calling number used for disambiguation of identification results
- delayed identity claim possible through name recognition
- consistency check of identified speaker and claimed identity
- identification not perceived by the caller

4.3.2. Caller-Identification for Call-Center: Caller-Black-List:

This fictitious application might offer some up-to-date information on a companies products. The informational service is not personalized and not security critical. It uses a VoiceXML-dialog to navigate to the desired information. Furthermore the service allows the caller to request transfer to an operator for questions not answered by the automatic service. The application uses an application scope grammar that recognizes calls for an operator.
Speaker identification is used to implement the following additional feature: The operators of this service can build up a restricted list of “special” customers, whose voiceprints are stored. If one of them is identified, a special strategy can be taken, for example one can get a high or a low priority.
Designation: The operator specifies a name for a newly enrolled speaker. No designation is needed for authentication.
Authentication: Speaker identification is performed on the utterance calling for the operator. The utterance is compared to all the enrolled voice prints.
Enrollment: While the operator is talking to the client, he can decide to add this client to the special list under a new or existing name. If the client was identified before the transfer, the operator can confirm the identification or not. If confirmed, the voiceprint will be adapted.

Classification:
- industry: telecom
- type: identification of specific individuals for call center
- security level: medium
- scale of deployment: national
- VoiceXML-Relevance: high
- Closest VoiceXML-Sample: 4.5

Features:
- almost same phrase for verification and enrollment (NL-Slot <operator>), but no phonetic consistency required
- no security issue
- no explicit verification or enrollment dialog
- caller not aware of speaker verification
- identification has application scope
- identification is done on all voice models in the speaker-verification database
- enrollment is done on buffered utterances
- existing voiceprints are adapted after confirmation

4.3.3. Speech-Skimming: Automatic Meeting Transcription

This is not a made up application. It was implemented as a research project and as such should in fact go into the real word applications section. See:
Florian Metze, Christian Fügen, Yue Pan, and Alex Waibel
Automatically Transcribing Meetings using Distant Microphones
In Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), Philadelphia, PA, USA, March 2005, IEEE.

Classification:
- industry: ?
- type: speech skimming
- security level: ?
- scale of deployment: small group

- VoiceXML-Relevance: low
- Closest VoiceXML-Sample: 4.4 and 4.6

Features:

- verification and enrollment on text-independent free speech
- no security issue
- on the fly enrollment
- speaker not aware of speaker verification
- identification has application scope
- identification is done on all voice models in the speaker-verification database
- enrollment is done on buffered utterances
- SIV used in combination with other independent voice processing, e.g. hot word detection, etc.

5. VoiceXML Pseudo Code Samples

This chapter includes selected sample application in VoiceXML-pseudo-code (Proposed SIV-extension). The SIV-extension tags are highlighted with turquoise color.

5.1. Example 3.1.1.A (Access to Mobile Phone Contract Details)

5.1.1. Enrollment Dialog

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!-- ********************************************************** -->
<!-- enrollment_proposal.jsp                                    -->
<!-- ********************************************************** -->

<vxml version="2.0" application="SIVSample/vxml/app_root.jsp"
xmlns="http://www.w3.org/2001/vxml" xml:lang="de-DE">

   <var name="promptpath" expr="'SIVSample/training/prompts/'"/>

    <!-- Populated at various stages, this variable holds the final result. -->
    <var name="outcome" expr="'rejected'"/>

   <link event="help">
     <grammar mode="voice" version="1.0" root="help">
       <rule id="help" scope="public"> help </rule>
     </grammar>
   </link>

   <link next="#Abort">
     <grammar mode="voice" version="1.0" root="abort">
      <rule id="abort" scope="public">
         <one-of>
          <item>abort</item>
          <item>exit</item>
          <item>cancel</item>
         </one-of>
      </rule>
     </grammar>
   </link>

   <form id="TrainIntro">
      <block>
          <prompt timeout="2s" bargein="true">
            <audio expr="promptpath + 'TrainingIntro.wav'"/>
            <audio expr="promptpath + 'PromptNew.wav'"/>
          </prompt>
         <goto next="#Train"/>
      </block>
   </form>

   <enrollment-form id="Train" key="session.connection.local.uri">

      <invalid>
         <if cond="application.lastresult$.interpretation.digits.length==4">
            <audio expr="promptpath + 'TrainPattern/OnlyFourDigits.wav'"/>
         <else/>
            <audio expr="promptpath + 'TrainPattern/InvalidUtterance.wav'"/>
         </if>
         <reprompt/>
      </invalid>

      <invalid count="4">
         <goto next = "#Abort"/>
      </invalid>

   <!--    the caller is asked to repeat the following phrases:   -->
   <!--    5 2 7 9 - 5 2 7 9                 -->
   <!--    6 3 1 4 - 6 3 1 4                 -->
   <!--    0 8 3 1 - 0 8 3 1                 -->
   <!--    7 0 9 8 - 7 0 9 8                 -->
   <!--    3 1 5 2 - 3 1 5 2                 -->
   <!--    1 4 0 8 - 1 4 0 8                  -->

      <field name="train_1" slot="digits" prompted_phrase="&lt;digits 52795279&gt;">
         <grammar src="${global.misc.grammars.SVEightDigit_String}"/>
```

```
        <grammar src="${global.misc.grammars.SVFourDigit_String}"/>

        <prompt bargein="true" timeout="2000ms">
           <audio expr="promptpath + '5.wav'"/>
           <audio expr="promptpath + '2.wav'"/>
           <audio expr="promptpath + '7.wav'"/>
           <audio expr="promptpath + '9f.wav'"/>
        </prompt>
     </field>

     <field name="train_2" slot="digits" prompted_phrase="&lt;digits 63146314&gt;">
        <grammar src="${global.misc.grammars.SVEightDigit_String}"/>
        <grammar src="${global.misc.grammars.SVFourDigit_String}"/>

        <prompt bargein="true" timeout="2000ms">
           <audio expr="promptpath + 'TrainPattern/Prompt.wav'"/>
           <audio expr="promptpath + '6.wav'"/>
           <audio expr="promptpath + '3.wav'"/>
           <audio expr="promptpath + '1.wav'"/>
           <audio expr="promptpath + '4f.wav'"/>
        </prompt>
     </field>

     <field name="train_3" slot="digits" prompted_phrase="&lt;digits 08310831&gt;">
        <grammar src="${global.misc.grammars.SVEightDigit_String}"/>
        <grammar src="${global.misc.grammars.SVFourDigit_String}"/>

        <prompt bargein="true" timeout="2000ms">
           <audio expr="promptpath + 'TrainPattern/Prompt.wav'"/>
           <audio expr="promptpath + '0.wav'"/>
           <audio expr="promptpath + '8.wav'"/>
           <audio expr="promptpath + '3.wav'"/>
           <audio expr="promptpath + '1f.wav'"/>
        </prompt>
     </field>

     <field name="train_4" slot="digits" prompted_phrase="&lt;digits 70987098&gt;">
        <grammar src="${global.misc.grammars.SVEightDigit_String}"/>
        <grammar src="${global.misc.grammars.SVFourDigit_String}"/>

        <prompt bargein="true" timeout="2000ms">
           <audio expr="promptpath + 'TrainPattern/Prompt.wav'"/>
           <audio expr="promptpath + '7.wav'"/>
           <audio expr="promptpath + '0.wav'"/>
           <audio expr="promptpath + '9.wav'"/>
           <audio expr="promptpath + '8f.wav'"/>
        </prompt>
     </field>

     <field name="train_5" slot="digits" prompted_phrase="&lt;digits 31523152&gt;">
        <grammar src="${global.misc.grammars.SVEightDigit_String}"/>
        <grammar src="${global.misc.grammars.SVFourDigit_String}"/>

        <prompt bargein="true" timeout="2000ms">
           <audio expr="promptpath + 'TrainPattern/Prompt.wav'"/>
           <audio expr="promptpath + '3.wav'"/>
           <audio expr="promptpath + '1.wav'"/>
           <audio expr="promptpath + '5.wav'"/>
           <audio expr="promptpath + '2f.wav'"/>
        </prompt>
     </field>

     <field name="train_6" slot="digits" prompted_phrase="&lt;digits 14081408&gt;">
        <grammar src="${global.misc.grammars.SVEightDigit_String}"/>
        <grammar src="${global.misc.grammars.SVFourDigit_String}"/>

        <prompt bargein="true" timeout="2000ms">
           <audio expr="promptpath + 'TrainPattern/Prompt.wav'"/>
           <audio expr="promptpath + '1.wav'"/>
           <audio expr="promptpath + '4.wav'"/>
           <audio expr="promptpath + '0.wav'"/>
           <audio expr="promptpath + '8f.wav'"/>
        </prompt>

        <filled>
           <audio expr="promptpath + 'TrainingOK.wav'"/>
           <submit next="SIVSample/servlet/CheckTrainingServlet" method="post"/>
        </filled>
     </field>
  </enrollment-form>
```

```
   <form id="Abort">
      <block>
         <audio expr="promptpath + 'TrainingFailed.wav'"/>
         <goto next="SIVSample/vxml/escapes/exit.jsp"/>
      </block>
   </form>

   <!-- document level event handlers -->
   <nomatch count="1">
      <audio expr="promptpath + 'TrainPattern/nomatch1.wav'"/>
      <reprompt/>
   </nomatch>
   <nomatch count="2">
      <audio expr="promptpath + 'TrainPattern/nomatch2.wav'"/>
      <reprompt/>
   </nomatch>
   <nomatch count="3">
      <audio expr="promptpath + 'TrainPattern/nomatch3.wav'"/>
      <reprompt/>
   </nomatch>
   <nomatch count="4">
      <goto next="#Abort"/>
   </nomatch>
   <noinput count="1">
      <audio expr="promptpath + 'TrainPattern/noinput1.wav'"/>
      <reprompt/>
   </noinput>
   <noinput count="2">
      <audio expr="promptpath + 'TrainPattern/noinput2.wav'"/>
      <reprompt/>
   </noinput>
   <noinput count="3">
      <audio expr="promptpath + 'TrainPattern/noinput3.wav'"/>
      <reprompt/>
   </noinput>
   <noinput count="4">
      <goto next="#Abort"/>
   </noinput>
   <catch event="help" count="1">
      <audio expr="promptpath + 'TrainPattern/help1.wav'"/>
      <reprompt/>
   </catch>
   <catch event="help" count="2">
      <audio expr="promptpath + 'TrainPattern/help2.wav'"/>
      <reprompt/>
   </catch>

</vxml>
```

5.1.2. Authentication Dialog

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!-- ********************************************************** -->
<!-- verify_proposal.jsp                                    -->
<!-- the caller is asked to repeat random digit sequences until a decision is reached -->
<!-- the dialog uses variable length verification -->
<!-- we need no manual adaptation, since automatic adaptation can
   be used for verification -->
<!-- ********************************************************** -->

<vxml version="2.0" application="SIVSample/vxml/app_root.jsp"
xmlns="http://www.w3.org/2001/vxml" xml:lang="de-DE">
   <property name="timeout" value="3s"/>
   <var name="promptpath" expr="'SIVSample/verification/prompts/'"/>

   <link event="help">
     <grammar mode="voice" version="1.0" root="help">
       <rule id="help" scope="public"> help </rule>
     </grammar>
   </link>

   <link next="#RETURN">
     <grammar mode="voice" version="1.0" root="abort">
      <rule id="abort" scope="public">
         <one-of>
          <item>abort</item>
          <item>exit</item>
```

```
            <item>cancel</item>
          </one-of>
       </rule>
     </grammar>
   </link>

   <form id="VerifyIntro">
      <block>
          <prompt timeout="2s" bargein="true">
            <audio expr="promptpath + 'VerificationIntro.wav'"/>
            <audio expr="promptpath + 'PromptNew.wav'"/>
          </prompt>
         <goto next="#Verify"/>
      </block>
   </form>

   <verification-form id="Verify" key="session.connection.local.uri">
      <!-- todo include leading zeros in sequence -->
      <var name="sequence" expr="new String(Math.round(10000*Math.random()))"/>

      <!-- form level event handlers -->

      <invalid>
         <if cond="verify.length==4">
            <audio expr="promptpath + 'VerifyPattern/OnlyFourDigits.wav'"/>
         <else/>
            <audio expr="promptpath + 'VerifyPattern/InvalidUtterance.wav'"/>
         </if>
         <reprompt/>
      </invalid>

      <invalid count="4">
         <goto next = "#RETURN"/>
      </invalid>

      <rejected>
         <audio expr="promptpath + 'VerificationFailed.wav'"/>
         <goto next="#RETURN"/>
      </rejected>

      <accepted>
         <audio expr="promptpath + 'VerificationPassed.wav'"/>
         <goto next="#RETURN"/>
      </accepted>

      <!-- this handler is only executed if the verification-form is left through a goto,
         but the rejected and accepted handlers are not executed -->

      <abandoned>
         <audio expr="promptpath + 'VerificationFailed.wav'"/>
      </abandoned>

      <!-- random digit sequences are verified until a decision is reached -->
      <field name="verify" slot="digits"
         prompted_phrase_expr="'&lt;digits ' + sequence + sequence +'&gt;'">
         <grammar src="${global.misc.grammars.SVEightDigit_String}"/>
         <grammar src="${global.misc.grammars.SVFourDigit_String}"/>
         <prompt>
            <audio expr="promptpath + sequence.charAt(0) + '.wav'"/>
            <audio expr="promptpath + sequence.charAt(1) + '.wav'"/>
            <audio expr="promptpath + sequence.charAt(2) + '.wav'"/>
            <audio expr="promptpath + sequence.charAt(3) + 'f.wav'"/>
         </prompt>
         <filled>
            <assign name="sequence" expr="String(Math.round(10000*Math.random()))"/>
            <goto nextitem="verify"/>
         </filled>
      </field>
   </verification-form>

   <form id="RETURN">
      <block>
         <submit next="SIVSample/servlet/verify.ResultServlet"
namelist="application.lastresult$.cumulative.decision"/>
      </block>
   </form>

   <!-- document level event handlers -->
   <nomatch count="1">
      <audio expr="promptpath + 'VerifyPattern/nomatch1.wav'"/>
      <reprompt/>
   </nomatch>
   <nomatch count="2">
      <audio expr="promptpath + 'VerifyPattern/nomatch2.wav'"/>
      <reprompt/>
   </nomatch>
   <nomatch count="3">
      <audio expr="promptpath + 'VerifyPattern/nomatch3.wav'"/>
      <reprompt/>
   </nomatch>
   <nomatch count="4">
      <goto next="#RETURN"/>
   </nomatch>

   <noinput count="1">
      <audio expr="promptpath + 'VerifyPattern/noinput1.wav'"/>
      <reprompt/>
   </noinput>
   <noinput count="2">
      <audio expr="promptpath + 'VerifyPattern/noinput2.wav'"/>
      <reprompt/>
   </noinput>
   <noinput count="3">
      <audio expr="promptpath + 'VerifyPattern/noinput3.wav'"/>
      <reprompt/>
   </noinput>
   <noinput count="4">
      <goto next="#RETURN"/>
   </noinput>

   <catch event="help" count="1">
      <audio expr="promptpath + 'VerifyPattern/help1.wav'"/>
      <reprompt/>
   </catch>
   <catch event="help" count="2">
      <audio expr="promptpath + 'VerifyPattern/help2.wav'"/>
      <reprompt/>
   </catch>

</vxml>
```

5.2. Example 3.1.1.B

In addition to the turquoise highlighting of the SIV-tags, the differences to the previous scenario are highlighted with yellow color.

5.2.1. Enrollment Dialog

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!-- *********************************************************** -->
<!-- enrollment_proposal.jsp                                         -->
<!-- *********************************************************** -->

<vxml version="2.0" application="SIVSample/vxml/app_root.jsp"
xmlns="http://www.w3.org/2001/vxml" xml:lang="de-DE">

   <var name="promptpath" expr="'SIVSample/training/prompts/'"/>

    <!-- Populated at various stages, this variable holds the final result. -->
    <var name="outcome" expr="'rejected'"/>

   <link event="help">
     <grammar mode="voice" version="1.0" root="help">
       <rule id="help" scope="public"> help </rule>
     </grammar>
   </link>

   <link next="#Abort">
     <grammar mode="voice" version="1.0" root="abort">
      <rule id="abort" scope="public">
         <one-of>
          <item>abort</item>
          <item>exit</item>
          <item>cancel</item>
         </one-of>
      </rule>
     </grammar>
   </link>

   <form id="TrainIntro">
      <block>
          <prompt timeout="2s" bargein="true">
            <audio expr="promptpath + 'TrainingIntro.wav'"/>
            <audio expr="promptpath + 'PromptNew.wav'"/>
          </prompt>
         <goto next="#Train"/>
      </block>
   </form>

<enrollment-form id="Train" key="'${param.VoiceID}'"
groupkey="session.connection.local.uri" usebuffer="svbuffer">

      <invalid>
         <if cond="application.lastresult$.interpretation.digits.length==4">
            <audio expr="promptpath + 'TrainPattern/OnlyFourDigits.wav'"/>
         <else/>
            <audio expr="promptpath + 'TrainPattern/InvalidUtterance.wav'"/>
         </if>
         <reprompt/>
      </invalid>

      <invalid count="4">
         <goto next = "#Abort"/>
      </invalid>

   <!--    the caller is asked to repeat the following phrases:   -->
   <!--    5 2 7 9 - 5 2 7 9                     -->
   <!--    6 3 1 4 - 6 3 1 4                     -->
   <!--    0 8 3 1 - 0 8 3 1                     -->
   <!--    7 0 9 8 - 7 0 9 8                     -->
   <!--    3 1 5 2 - 3 1 5 2                     -->
   <!--    1 4 0 8 - 1 4 0 8                      -->

      <field name="train_1" slot="digits" prompted_phrase="&lt;digits 52795279&gt;">
         <grammar src="${global.misc.grammars.SVEightDigit_String}"/>
         <grammar src="${global.misc.grammars.SVFourDigit_String}"/>

         <prompt bargein="true" timeout="2000ms">
            <audio expr="promptpath + '5.wav'"/>
            <audio expr="promptpath + '2.wav'"/>
            <audio expr="promptpath + '7.wav'"/>
            <audio expr="promptpath + '9f.wav'"/>
         </prompt>
      </field>

      <field name="train_2" slot="digits" prompted_phrase="&lt;digits 63146314&gt;">
         <grammar src="${global.misc.grammars.SVEightDigit_String}"/>
         <grammar src="${global.misc.grammars.SVFourDigit_String}"/>

         <prompt bargein="true" timeout="2000ms">
            <audio expr="promptpath + 'TrainPattern/Prompt.wav'"/>
            <audio expr="promptpath + '6.wav'"/>
            <audio expr="promptpath + '3.wav'"/>
            <audio expr="promptpath + '1.wav'"/>
            <audio expr="promptpath + '4f.wav'"/>
         </prompt>
      </field>

      <field name="train_3" slot="digits" prompted_phrase="&lt;digits 08310831&gt;">
         <grammar src="${global.misc.grammars.SVEightDigit_String}"/>
         <grammar src="${global.misc.grammars.SVFourDigit_String}"/>

         <prompt bargein="true" timeout="2000ms">
            <audio expr="promptpath + 'TrainPattern/Prompt.wav'"/>
            <audio expr="promptpath + '0.wav'"/>
            <audio expr="promptpath + '8.wav'"/>
            <audio expr="promptpath + '3.wav'"/>
            <audio expr="promptpath + '1f.wav'"/>
         </prompt>
      </field>

      <field name="train_4" slot="digits" prompted_phrase="&lt;digits 70987098&gt;">
         <grammar src="${global.misc.grammars.SVEightDigit_String}"/>
         <grammar src="${global.misc.grammars.SVFourDigit_String}"/>
         <prompt bargein="true" timeout="2000ms">
            <audio expr="promptpath + 'TrainPattern/Prompt.wav'"/>
            <audio expr="promptpath + '7.wav'"/>
            <audio expr="promptpath + '0.wav'"/>
            <audio expr="promptpath + '9.wav'"/>
            <audio expr="promptpath + '8f.wav'"/>
         </prompt>
      </field>

      <field name="train_5" slot="digits" prompted_phrase="&lt;digits 31523152&gt;">
         <grammar src="${global.misc.grammars.SVEightDigit_String}"/>
         <grammar src="${global.misc.grammars.SVFourDigit_String}"/>

         <prompt bargein="true" timeout="2000ms">
            <audio expr="promptpath + 'TrainPattern/Prompt.wav'"/>
            <audio expr="promptpath + '3.wav'"/>
            <audio expr="promptpath + '1.wav'"/>
            <audio expr="promptpath + '5.wav'"/>
            <audio expr="promptpath + '2f.wav'"/>
         </prompt>
      </field>

      <field name="train_6" slot="digits" prompted_phrase="&lt;digits 14081408&gt;">
         <grammar src="${global.misc.grammars.SVEightDigit_String}"/>
         <grammar src="${global.misc.grammars.SVFourDigit_String}"/>

         <prompt bargein="true" timeout="2000ms">
            <audio expr="promptpath + 'TrainPattern/Prompt.wav'"/>
            <audio expr="promptpath + '1.wav'"/>
            <audio expr="promptpath + '4.wav'"/>
            <audio expr="promptpath + '0.wav'"/>
            <audio expr="promptpath + '8f.wav'"/>
         </prompt>

         <filled>
            <audio expr="promptpath + 'TrainingOK.wav'"/>
            <submit next="SIVSample/servlet/CheckTrainingServlet" method="post"/>
         </filled>
      </field>
   </enrollment-form>

   <form id="Abort">
      <block>
         <audio expr="promptpath + 'TrainingFailed.wav'"/>
         <goto next="SIVSample/vxml/escapes/exit.jsp"/>
      </block>
   </form>

   <!-- document level event handlers -->
   <nomatch count="1">
      <audio expr="promptpath + 'TrainPattern/nomatch1.wav'"/>
      <reprompt/>
   </nomatch>
   <nomatch count="2">
      <audio expr="promptpath + 'TrainPattern/nomatch2.wav'"/>
      <reprompt/>
   </nomatch>
   <nomatch count="3">
      <audio expr="promptpath + 'TrainPattern/nomatch3.wav'"/>
      <reprompt/>
   </nomatch>
   <nomatch count="4">
      <goto next="#Abort"/>
   </nomatch>
   <noinput count="1">
      <audio expr="promptpath + 'TrainPattern/noinput1.wav'"/>
      <reprompt/>
   </noinput>
   <noinput count="2">
      <audio expr="promptpath + 'TrainPattern/noinput2.wav'"/>
      <reprompt/>
   </noinput>
   <noinput count="3">
      <audio expr="promptpath + 'TrainPattern/noinput3.wav'"/>
      <reprompt/>
   </noinput>
   <noinput count="4">
      <goto next="#Abort"/>
   </noinput>
   <catch event="help" count="1">
      <audio expr="promptpath + 'TrainPattern/help1.wav'"/>
      <reprompt/>
   </catch>
   <catch event="help" count="2">
      <audio expr="promptpath + 'TrainPattern/help2.wav'"/>
      <reprompt/>
   </catch>

</vxml>
```

5.2.2. Authentication Dialog

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!-- ********************************************************** -->
<!-- verify_proposal.jsp                                     -->
<!-- the caller is asked to repeat random digit sequences until a decision is reached -->
<!-- the dialog uses variable length verification -->
<!-- we need no manual adaptation, since automatic adaptation can
   be used for verification -->
<!-- ********************************************************** -->

<vxml version="2.0" application="SIVSample/vxml/app_root.jsp"
xmlns="http://www.w3.org/2001/vxml" xml:lang="de-DE">
   <property name="timeout" value="3s"/>
   <var name="promptpath" expr="'SIVSample/verification/prompts/'"/>

   <link event="help">
     <grammar mode="voice" version="1.0" root="help">
       <rule id="help" scope="public"> help </rule>
     </grammar>
   </link>

   <link next="#RETURN">
     <grammar mode="voice" version="1.0" root="abort">
      <rule id="abort" scope="public">
         <one-of>
          <item>abort</item>
          <item>exit</item>
          <item>cancel</item>
         </one-of>
      </rule>
     </grammar>
   </link>

   <form id="VerifyIntro">
      <block>
          <prompt timeout="2s" bargein="true">
            <audio expr="promptpath + 'VerificationIntro.wav'"/>
            <audio expr="promptpath + 'PromptNew.wav'"/>
          </prompt>
         <goto next="#Verify"/>
      </block>
   </form>

   <verification-form id="Verify" key="session.connection.local.uri">
      <!-- todo include leading zeros in sequence -->
      <var name="sequence" expr="new String(Math.round(10000*Math.random()))"/>

      <!-- form level event handlers -->

      <invalid>
         <if cond="verify.length==4">
            <audio expr="promptpath + 'VerifyPattern/OnlyFourDigits.wav'"/>
         <else/>
            <audio expr="promptpath + 'VerifyPattern/InvalidUtterance.wav'"/>
         </if>
         <reprompt/>
      </invalid>

      <invalid count="4">
         <goto next = "#RETURN"/>
      </invalid>

      <rejected>
          <assignbuffer name="svbuffer"/>
          <submit next="SIVSample/servlet/identify.Rejected"/>
      </rejected>
      <accepted>
         <adapt cond="application.lastresult$.numvalidutterances > 1"/>
         <audio expr="promptpath + 'VerificationPassed.wav'"/>
         <goto next="#RETURN"/>
      </accepted>

      <!-- this handler is only executed if the verification-form is left through a goto,
         but the rejected and accepted handlers are not executed -->

      <abandoned>
         <audio expr="promptpath + 'VerificationFailed.wav'"/>
      </abandoned>

      <!-- random digit sequences are verified until a decision is reached -->
      <field name="verify" slot="digits"
         prompted_phrase_expr="'&lt;digits ' + sequence + sequence +'&gt;'">
         <grammar src="${global.misc.grammars.SVEightDigit_String}"/>
         <grammar src="${global.misc.grammars.SVFourDigit_String}"/>
         <prompt>
            <audio expr="promptpath + sequence.charAt(0) + '.wav'"/>
            <audio expr="promptpath + sequence.charAt(1) + '.wav'"/>
            <audio expr="promptpath + sequence.charAt(2) + '.wav'"/>
            <audio expr="promptpath + sequence.charAt(3) + 'f.wav'"/>
         </prompt>
         <filled>
            <assign name="sequence" expr="String(Math.round(10000*Math.random()))"/>
            <goto nextitem="verify"/>
         </filled>
      </field>
   </identification-form>

   <form id="RETURN">
      <block>
         <submit next="SIVSample/servlet/verify.ResultServlet"
namelist="application.lastresult$.cumulative.decision"/>
      </block>
   </form>

   <!-- document level event handlers -->
   <nomatch count="1">
      <audio expr="promptpath + 'VerifyPattern/nomatch1.wav'"/>
      <reprompt/>
   </nomatch>
   <nomatch count="2">
      <audio expr="promptpath + 'VerifyPattern/nomatch2.wav'"/>
      <reprompt/>
   </nomatch>
   <nomatch count="3">
      <audio expr="promptpath + 'VerifyPattern/nomatch3.wav'"/>
      <reprompt/>
   </nomatch>
   <nomatch count="4">
      <goto next="#RETURN"/>
   </nomatch>

   <noinput count="1">
      <audio expr="promptpath + 'VerifyPattern/noinput1.wav'"/>
      <reprompt/>
   </noinput>
   <noinput count="2">
      <audio expr="promptpath + 'VerifyPattern/noinput2.wav'"/>
      <reprompt/>
   </noinput>
   <noinput count="3">
      <audio expr="promptpath + 'VerifyPattern/noinput3.wav'"/>
      <reprompt/>
   </noinput>
   <noinput count="4">
      <goto next="#RETURN"/>
   </noinput>

   <catch event="help" count="1">
      <audio expr="promptpath + 'VerifyPattern/help1.wav'"/>
      <reprompt/>
   </catch>
   <catch event="help" count="2">
      <audio expr="promptpath + 'VerifyPattern/help2.wav'"/>
      <reprompt/>
   </catch>
```

```
</vxml>
```

5.3. Example 3.1.1.C

The enrollment dialog stays the same. In addition to the turquoise highlighting of the SIV-tags, the differences to the previous scenario are highlighted with yellow color.

5.3.1. Authentication Dialog

```
<?xml version="1.0" encoding="iso-8859-1"?>

<vxml version="2.0" application="SIVSample/vxml/app_root.jsp"
xmlns="http://www.w3.org/2001/vxml" xml:lang="de-DE">
   <var name="promptpath" expr="'SIVSample/verification/prompts/'"/>
   <var name="identity" expr="'${param.VoiceID}'"/>
   <var name="numbuffered" expr="0"/>

   <link event="help">
     <grammar mode="voice" version="1.0" root="help">
       <rule id="help" scope="public"> help </rule>
     </grammar>
   </link>

   <link next="#RETURN">
     <grammar mode="voice" version="1.0" root="abort">
      <rule id="abort" scope="public">
         <one-of>
          <item>abort</item>
          <item>exit</item>
          <item>cancel</item>
         </one-of>
      </rule>
     </grammar>
   </link>

   <form id="CheckUserId">
      <filled>
         <assign name="identity" expr="callerId"/>
      </filled>

      <block cond="identity!=''">
         <goto next="#VerifyIntro"/>
      </block>

      <field name="callerId" slot="digits">
         <grammar src="${global.misc.grammars.PhoneNumber}"/>
          <prompt timeout="2s" bargein="true">
            <audio expr="promptpath + 'PhonePrompt.wav'"/>
          </prompt>
      </field>

      <field name="confirmation">
         <grammar src="${global.misc.grammars.YesNo}"/>
         <prompt>
            <audio expr="promptpath + 'ConfirmIntro.wav'"/>
            <foreach item="digit" array="callerId">
               <audio expr="promptpath + digit + '.wav'"/>
            </foreach>
         </prompt>
         <filled>
            <if cond="confirmation=='no'">
               <clear namelist="callerId confirmation"/>
            <else/>
               <assign name="numbuffered" expr="1"/>
            </if>
         </filled>
      </field>
   </form>

   <form id="VerifyIntro">
      <block>
          <prompt timeout="2s" bargein="true">
            <audio expr="promptpath + 'VerificationIntro.wav'"/>
            <audio expr="promptpath + 'PromptNew.wav'"/>
          </prompt>
         <goto next="#Verify"/>
      </block>
   </form>

   <identification-form id="Verify" key="identity" scope="document"
       siv-condition="&lt;digits&gt;" minutterances="1+numbuffered"
       maxutterances="3+numbuffered">

      <!-- todo include leading zeros in sequence -->
      <var name="sequence" expr="new String(Math.round(10000*Math.random()))"/>

      <nonexistent>
         <!-- a variable of name "svbuffer" must be declared in the
            app_root document -->

         <assignbuffer name="svbuffer"/>
         <submit next="SIVSample/servlet/training.GetTANList" namelist="identity"/>
      </nonexistent>

      <invalid>
         <if cond="verify.length==4">
            <audio expr="promptpath + 'VerifyPattern/OnlyFourDigits.wav'"/>
         <else/>
            <audio expr="promptpath + 'VerifyPattern/InvalidUtterance.wav'"/>
         </if>
         <reprompt/>
      </invalid>

      <invalid count="4">
         <goto next = "#RETURN"/>
      </invalid>

      <rejected>
         <assignbuffer name="svbuffer"/>
         <submit next="SIVSample/servlet/identify.Rejected"/>
      </rejected>

      <accepted>
         <if cond="application.lastresult$.incremental.decision!='rejected'>
            <adapt cond="application.lastresult$.numvalidutterances > 1"/>
            <audio expr="promptpath + 'VerificationPassed.wav'"/>
            <goto next="#RETURN"/>
         </if>
      </accepted>

      <!-- this handler is only executed if the verification-form is left through a goto,
         but the rejected and accepted handlers are not executed -->

      <abandoned>
         <audio expr="promptpath + 'VerificationFailed.wav'"/>
      </abandoned>

      <!-- random digit sequences are verified until a decision is reached -->
      <field name="verify" slot="digits"
         prompted_phrase_expr="'&lt;digits ' + sequence + sequence +'&gt;'">
         <grammar src="${global.misc.grammars.SVEightDigit_String}"/>
         <grammar src="${global.misc.grammars.SVFourDigit_String}"/>
         <prompt bargein="true" timeout="2000ms">
            <audio expr="promptpath + sequence.charAt(0) + '.wav'"/>
            <audio expr="promptpath + sequence.charAt(1) + '.wav'"/>
            <audio expr="promptpath + sequence.charAt(2) + '.wav'"/>
            <audio expr="promptpath + sequence.charAt(3) + 'f.wav'"/>
         </prompt>
         <filled>
            <assign name="sequence" expr="String(Math.round(10000*Math.random()))"/>
            <goto nextitem="verify"/>
         </filled>
      </field>
   </identification-form>

   <form id="RETURN">
      <block>
         <submit next="SIVSample/servlet/verify.ResultServlet"
namelist="application.lastresult$.cumulative.decision"/>
      </block>
   </form>

   <!-- document level event handlers -->
   <nomatch count="1">
      <audio expr="promptpath + 'VerifyPattern/nomatch1.wav'"/>
      <reprompt/>
   </nomatch>
   <nomatch count="2">
      <audio expr="promptpath + 'VerifyPattern/nomatch2.wav'"/>
      <reprompt/>
   </nomatch>
   <nomatch count="3">
      <audio expr="promptpath + 'VerifyPattern/nomatch3.wav'"/>
      <reprompt/>
   </nomatch>
   <nomatch count="4">
      <goto next="#RETURN"/>
   </nomatch>

   <noinput count="1">
      <audio expr="promptpath + 'VerifyPattern/noinput1.wav'"/>
      <reprompt/>
   </noinput>
   <noinput count="2">
      <audio expr="promptpath + 'VerifyPattern/noinput2.wav'"/>
      <reprompt/>
   </noinput>
   <noinput count="3">
      <audio expr="promptpath + 'VerifyPattern/noinput3.wav'"/>
      <reprompt/>
   </noinput>
   <noinput count="4">
      <goto next="#RETURN"/>
   </noinput>

   <catch event="help" count="1">
      <audio expr="promptpath + 'VerifyPattern/help1.wav'"/>
      <reprompt/>
   </catch>
   <catch event="help" count="2">
      <audio expr="promptpath + 'VerifyPattern/help2.wav'"/>
      <reprompt/>
   </catch>

</vxml>
```

5.4. Example 4.1.2 (Multi-Modal ATM-access)

This dialog represents the authentication part of application 4.1.2. This implementation asks for voice input first and for fingerprint afterwards.

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!-- ********************************************************** -->
<!-- multimode_verification.jsp                                 -->
<!-- ********************************************************** -->

<vxml version="2.0" application="SIVSample/vxml/app_root.jsp"
xmlns="http://www.w3.org/2001/vxml" xml:lang="de-DE">
   <var name="promptpath" expr="'SIVSample/verification/prompts/'"/>
   <var name="outcome" expr="'rejected'"/>

   <form id="VerifyIntro">
      <block>
          <prompt>
            <audio expr="promptpath + 'VerificationIntro.wav'"/>
          </prompt>
         <goto next="#VerifyVoice"/>
      </block>
   </form>

   <verification-form id="VerifyVoice" key="'${param.VoiceID}'" text-
dependent="true">
      <!—- text-dependant implies phonetic consistency check with stored pass
phrase -->
      <!-- form level event handlers -->
      <rejected>
         <audio expr="promptpath + 'VoiceVerificationFailed.wav'"/>
         <assign name="outcome" expr="'voicerejected'"/>
         <goto next="#RETURN"/>
      </rejected>

      <accepted>
         <audio expr="promptpath + 'VoiceVerificationPassed.wav'"/>
         <goto next="#VerifyFingerprint"/>
      </accepted>

      <field name="verify">
         <prompt>
            <audio expr="promptpath + 'PromptSecretPhrase.wav'"/>
         </prompt>

         <nomatch>
            <audio expr="promptpath + 'SecretPhrase/nomatch.wav'"/>
            <reprompt/>
         </nomatch>
         <nomatch count="4">
            <goto next="#RETURN"/>
         </nomatch>

         <noinput>
            <audio expr="promptpath + 'SecretPhrase/noinput.wav'"/>
            <reprompt/>
         </noinput>
         <noinput count="4">
            <goto next="#RETURN"/>
         </noinput>
      </field>
   </verification-form>

   <verification-form id="VerifyFingerprint" key="'${param.VoiceID}'"
      mode="fingerprint">
      <!-- form level event handlers -->
      <rejected>
         <audio expr="promptpath + 'FingerprintVerificationFailed.wav'"/>
         <assign name="outcome" expr="'fingerprintrejected'"/>
         <goto next="#RETURN"/>
      </rejected>

      <accepted>
         <audio expr="promptpath + 'VerificationPassed.wav'"/>
         <assign name="outcome" expr="'accepted'"/>
         <goto next="#RETURN"/>
      </accepted>

      <field name="verify">
         <prompt>
            <audio expr="promptpath + 'PromptFingerprint.wav'"/>
         </prompt>
      </field>
   </verification-form>

   <form id="RETURN">
      <block>
         <submit next="SIVSample/servlet/verify.ResultServlet"
namelist="outcome"/>
      </block>
   </form>
</vxml>
```

5.5. Example 4.3.2 (Caller-Black-List):

5.5.1. Application Root Document

```
<?xml version="1.0" encoding="iso-8859-1"?>

<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml" xml:lang="de-DE">
```

```
   <var name="svbuffer"/>
   <!-- identification session with application scope -->
   <identification-form id="special" scope="document">
      <grammar scope="document" mode="voice" version="1.0" root="operator">
         <rule id="operator" scope="public"> operator </rule>
      </grammar>

      <field name="operator" slot="operator"/>

      <filled>
         <assignbuffer name="svbuffer"/>
      </filled>

      <accepted>
      <!-- if a special customer was identified goto servlet that decides
         which transfer to call -->
         <submit next="SIVSample/servlet/OperatorSpecialFound"
            namelist="application.lastresult$.identifiedspeaker"/>
      </accepted>

      <rejected>
         <goto next="#transfer"/>
      </rejected>
   </identification-form>

   <form id="transfer">
      <transfer name="talk2operator" dest="tel:111" bridge="true"/>

      <!-- goto servlet that checks if the operator decided to add
         this client to the special list -->
      <block>
         <submit next="SIVSample/servlet/OperatorSpecialNew"/>
      </block>
   </form>
</vxml>
```

5.5.2. Enrollment Dialog

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!-- ********************************************************** -->
<!-- enrollspecial.jsp                                        -->
<!-- ********************************************************** -->

<!-- this is called after a successful operator transfer to enroll a speaker
into
   the special list or to adapt a confirmed speakers voiceprint
   -->
<vxml version="2.0" application="SIVSample/vxml/app_root_proposal.jsp"
xmlns="http://www.w3.org/2001/vxml" xml:lang="de-DE">

   <enrollment-form id="Train" key="${param.VoiceID}"
      use-buffer="special.buffer" writemode="adapt">
      <block>
         <if cond="talk2operator=='far_end_disconnect'">
            <goto next="SIVSample/InfoApp/MainMenu.vxml"/>
         </if>
      </block>
   </enrollment-form>
</vxml>
```

5.6. Example 4.1.3 (PIN-less Call Center Authentication)

This code represents only a simplified version of the application described in 4.1.3. The disambiguation part is left out in this examle code.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- ********************************************************** -->
<!-- freespeech.jsp                                         -->
<!-- ********************************************************** -->

<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml" xml:lang="de-DE">
   <var name="svbuffer"/>
   <var name="newspeakerkey"/>
   <!-- identification with no key specified means all voiceprints
       are considered -->
   <identification-form>
      <!-- the transfer is the only input element -->
      <transfer name="talk2operator" dest="tel:111" bridge="true"/>

      <!-- in this case the decision from the verifier can come before the input element
is filled -->

      <!-- call a servlet to display the identified speaker to the operator. The FIA
remains in the input state. -->
      <accepted>
         <data src="SIVSample/servlet/SpeakerFound"
            namelist="special.identifiedspeaker"/>
      </accepted>

      <!-- call a servlet that tells the operator that
         the caller is not known.
         Afterwards FIA is still in the input state.
          -->
      <rejected>
         <data src="SIVSample/servlet/UnknownSpeaker"/>
         <assignbuffer name="svbuffer"/>
      </rejected>

      <filled>
         <!-- call a servlet to fetch the new speakers name, if the operator entered one.
Then store the voicemodel -->
         <if cond="application.lastresult$.decision=='rejected'/>
            <data name="newspeaker" src="SIVSample/servlet/GetNewSpeaker"/>
            <if cond="newspeaker!=''"/>
               <assign name="newspeakerkey" expr="newspeaker"/>
               <goto next="#enrollment"/>
            </if>
         </if>
      </filled>
   </identification-form>

   <enrollment-form id="enrollment" usebuffer="svbuffer" key="newspeakerkey"/>
</vxml>
```

6. References

[Requirements] “Speaker Verification and Identification Requirements Document” by VXML Biometrics Special Interest Group (BioSIG)

[Best Practices] “Speaker Verification and Identification Introduction and Best Practices Document” by VXML Biometrics Special Interest Group (BioSIG)