



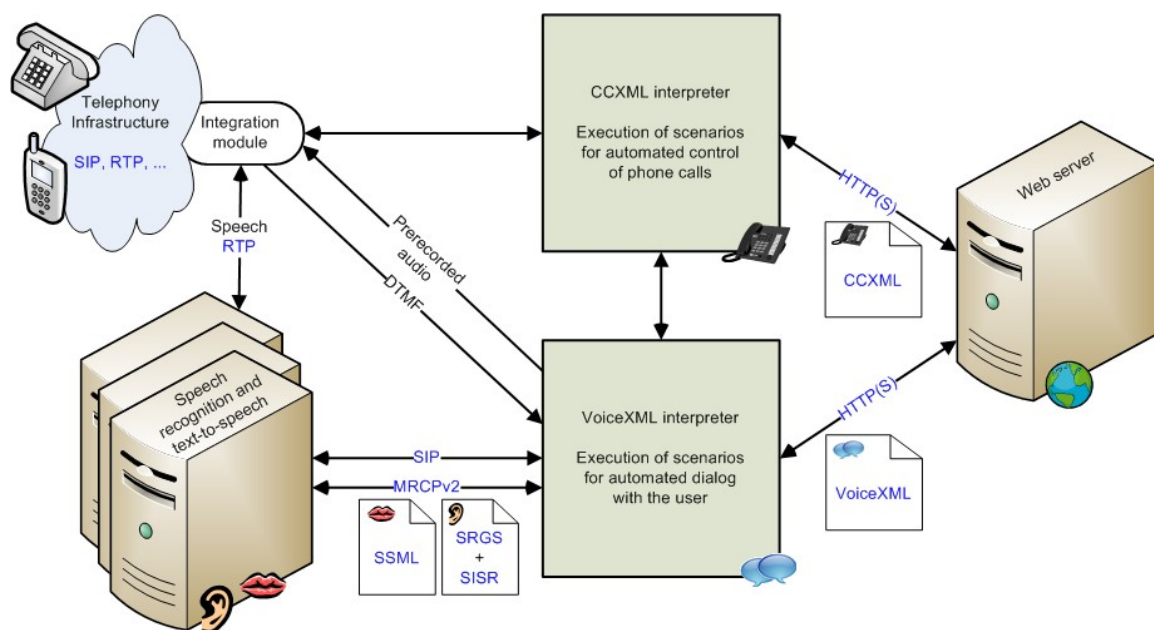
# OptimTalk Voice Browser Integration with PBX and IT Systems

**OptimSys, s.r.o.** tel.: +420 541 143 065  
U Vodárny 2 fax: +420 541 143 066  
61600 Brno info@optimsys.com  
Czech Republic www.optimsys.com

OptimTalk Voice Browser is a software platform for running telephony and speech applications such as interactive voice response (IVR) applications. It is composed of two main functional parts:

- A CCXML interpreter that executes scenarios for automatic control of phone calls described by means of the CCXML language.
- A VoiceXML interpreter that executes communication scenarios controlling the dialog between a user and the system (in the simplest case they can have the form of today's common IVR menus navigated with phone buttons).

The OptimTalk Voice Browser scheme is depicted in Figure 1.



*Figure 1. OptimTalk Voice Browser Scheme*

## Integration with PBX Systems

The CCXML interpreter is independent of the underlying telephony infrastructure. It is connected with a PBX system through an integration module which is tailored for every PBX system or hardware telephony cards application interface. OptimSys abandoned the development of integration modules tailored for proprietary or legacy application interfaces such as TAPI and Global Call API and fully concentrates on integration using the standard SIP protocol today. The connection to PBX systems that do not support SIP is realized by using third-party SIP gateways.

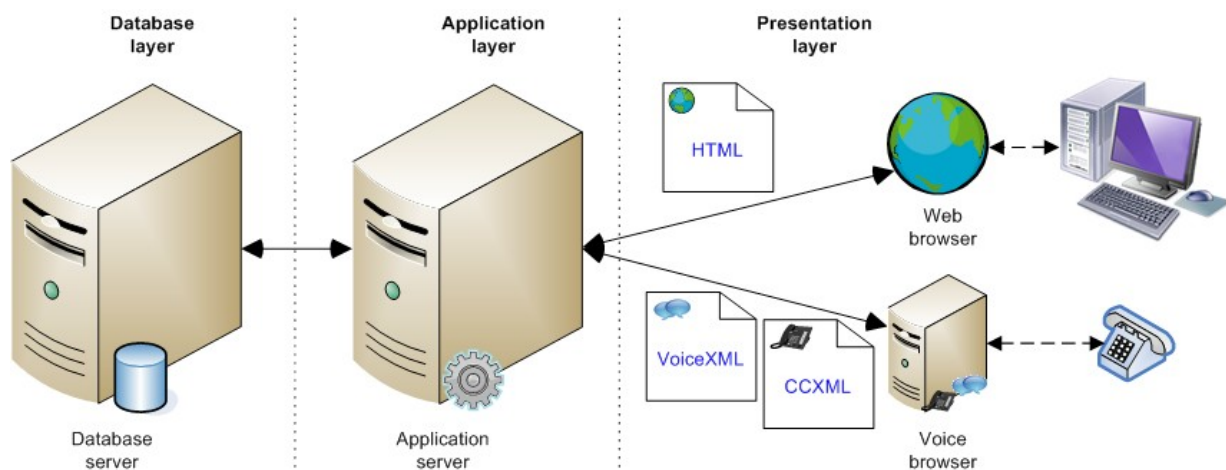
The CCXML interpreter gets information about current traffic in the telephone network through the integration module, such as information about incoming calls, notifications about picking up and hanging up a call, etc. In the opposite direction, the CCXML interpreter instructs the PBX system through the integration module to perform various commands such as to establish an outgoing call, to answer/reject an incoming call, etc.

## Integration with IT Systems

Both VoiceXML and CCXML scenarios can be dynamically generated and modified according to the current data in customer's IT systems. They can also store data to these systems. The integration and interaction with IT systems can be realized in several ways.

### Integration with Systems Based on the Three-Tier Architecture

This is the most typical integration method which is convenient especially for information systems that make their data available through a web interface, which is nowadays a majority of them. These systems are mostly based on the standard three-tier architecture. Integration of the OptimTalk Voice Browser to the three-tier architecture is shown in Figure 2.



**Figure 2. OptimTalk Voice Browser in the three-tier architecture**

The above mentioned systems usually constitute of a database server and an application server. The user typically sends requests and data to the application server via the HTTP protocol using a web browser and the application server sends responses back to the web browser in the form of HTML pages. The web browser then displays these pages to the user. The HTML pages are generated by the application server according to the current data taken from the database server.

Integration of the OptimTalk Voice Browser with this type of applications is usually straightforward. The application server must be extended so that it is able to generate, besides the HTML pages, also respective VoiceXML and CCXML scenarios. The entire application logic and data in the database are kept unchanged.

When the user accesses the system by voice, he/she uses a phone instead of a keyboard and monitor and the web browser's role is taken over by the OptimTalk Voice Browser. It sends requests and data to the application server via the HTTP protocol and receives responses back from the application server in the form of CCXML and VoiceXML scenarios. These scenarios describe the way of how to present information to the user by voice and how to interact with the user. The OptimTalk Voice Browser basically offers another communication channel to the information system in this case.

A use case scenario may look as follows:

1. The user calls given phone number.

2. The CCXML interpreter is informed about a new incoming call and receives additional information about the call from the PBX system, especially the caller's phone number.
3. The CCXML interpreter sends a request to the application server to generate a CCXML scenario for servicing the incoming call. The caller's phone number is part of the request.
4. The application server looks given phone number up in the database and finds out that the caller is a registered user that has an identification number assigned. The application server generates a CCXML scenario for servicing this call, inserts user's identification number into it and sends it back to the CCXML interpreter.
5. The CCXML interpreter proceeds according to the generated scenario, for example it answers the call and transfers the user to an interactive voice response (IVR) system controlled by the VoiceXML interpreter. It also passes the user's identification number along to the VoiceXML interpreter.
6. The VoiceXML interpreter sends a request to the application server to generate a VoiceXML scenario for interaction with this particular user. The user's identification number is included in the request.
7. Based on the identification number, the application server looks up in the database that the user is a premium service subscriber, generates an appropriate VoiceXML scenario and sends it back to the VoiceXML interpreter.
8. The VoiceXML interpreter interacts with the user according to this scenario. The scenario may contain instructions for eliciting some information from the user. This information can be sent to the application server that generates a new scenario on its basis which describes how to continue the interaction. Thus the interaction can be influenced by current data provided by the user.
9. The user hangs up.

### **Loading Data from the Systems into a Pre-generated Scenario**

VoiceXML as well as CCXML offer means that may reduce the number of round trips to the application server and/or reduce the amount of transferred data. Instead of generating a whole new scenario, it is possible to only transfer fresh data from the database to the current scenario that subsequently processes them. A suitable representation of the data for this purpose is the standard XML format.

This technique is a parallel to the AJAX technology used in web browsers for modern web applications with rich user interfaces (a.k.a. Web 2.0).

### **Communication with the Systems Using Web Services**

Thanks to the boom of web services and service-oriented architecture based systems, standardized SOAP protocol gained a lot of popularity. Although the VoiceXML and CCXML standards include no SOAP protocol support, OptimTalk Voice Browser offers an extension that enables SOAP based communication. It is possible to call web services right away from VoiceXML as well as CCXML scenarios and influence subsequent scenario execution by their results.

Since more and more systems are making their functions accessible through the web services, the importance of the SOAP protocol is rising. Using web services, one can for example read informa-

tion from all sorts of databases and registers, check weather forecast, get information about cultural events or travel agencies offers, search in maps, send SMSs and a lot more.

### **Other Communication Possibilities with the Systems**

There are two approaches to solve the situation when it is not possible to use either the application server services or web services for communication with third party systems.

The first approach lies in creation of a separate application on the web server which accesses the database directly (e.g. through the ODBC interface) and sends obtained results back to the VoiceXML or CCXML interpreter in an appropriate format (e.g. XML). The advantage of this approach is in keeping the standard three-tier architecture, except that the application server is worked around.

The second approach uses a proprietary extension of OptimTalk Voice Browser's VoiceXML and CCXML interpreters that makes it possible to call a function located in an external dynamic library written in an arbitrary programming language (C/C++, Pascal, Visual Basic, etc.). This library can also access the database directly and send obtained data back to the respective interpreter. A disadvantage of this approach is that it is not a standard solution.

On the other hand, an advantage lies in the fact that the dynamic library does not have to implement database access only, but can also perform any other operation. This allows for integration of the OptimTalk Voice Browser with various non-standard and special systems.

### **Outgoing Call Initiation from CRM Systems (Click to Dial)**

The customers often require the possibility to call a client with just a single click in their CRM system. The OptimTalk Voice Browser offers an elegant solution for this requirement. There is a simple web server embedded in the CCXML interpreter that serves for receiving external events through the standard HTTP or HTTPS protocol. Incoming events might lead for example to execution of a defined CCXML scenario.

To initiate an outgoing call right from a CRM system it is then enough to put a link with proper parameters into the CRM system referencing this built-in web server that arranges everything using a suitable CCXML scenario.