



Migrating Legacy Applications to VoiceXML with Voxeo

Migrating IVR Applications to a 100% VoiceXML compliant platform – and why it's a good idea

August 2010

Table of Contents

1	Introduction	2
2	About VoiceObjects.....	2
3	Speeding migration to VoiceXML with Voxeo	5
	Development Productivity	5
	VoiceObjectsXML.....	6
	Application code.....	11
	Other Resources.....	11
4	Migration lessons learned	12
5	Voxeo Unlocked Communications	13
6	Learn More.....	14

1 Introduction

Many enterprises and voice application providers are facing IVR platform end-of-life. With most vendors offering dialog design tools that lock customers into their own platform, platform end-of-life usually means application end-of-life too. VoiceXML is now a mature and well-adopted standard in the telephony and IVR industry – and the top target format for voice application migration today. But what is the best way to move your proprietary legacy applications to VoiceXML? And more importantly, how do you ensure you won't be locked in again?

This whitepaper demonstrates how Voxeo can help you:

- Speed up migration to VoiceXML, as well as the migration of hand-coded VoiceXML to a superior VoiceXML-based application environment
- Develop applications using a platform-independent service creation environment that protects your investment with automatic portability to over 30 leading IVR platforms, including but not limited to Voxeo's own Prophecy IVR
- Avoid the lock-in pitfall by using a platform that is 100% standards compliant and unlocked at every layer – enabling your solution to easily evolve with the needs of your business

Voxeo VoiceObjects provides service creation, execution, management, reporting and analytics capabilities that are unmatched in the industry. While companies understand the value of having a comprehensive, mature product for their voice application endeavors, they fear being locked in to a single vendor. Unlike other tools, VoiceObjects provides an open and flexible environment that eliminates vendor lock-in at every level. We'll explain how you can use this powerful tool to simplify the migration of your existing applications and discuss how both VoiceObjects and the Voxeo Prophecy IVR platform are designed to protect your investment and put you in control.

2 About VoiceObjects

Voxeo VoiceObjects is a carrier-grade Application Lifecycle Suite for Unified Self-Service. The solution is based on open standards such as VoiceXML, HTTP, (X)HTML, XML, SQL, and SIP. Using VoiceObjects, developers can deploy one service definition to interact with customers over multiple channels:

- Voice
- Video

- (Mobile) Web
- Short Message Service (SMS)
- Instant Messaging (IM, “chat bots”)
- Unstructured Supplementary Services Data (USSD)
- Smartphone Apps

This is realized through VoiceObjects’ Layer concept, which allows the creation and maintenance of services that share application logic and backend connectors, but have different presentation layers. The same concept is used to support highly personalized applications, such as multi-lingual self-service or applications that adapt to caller preferences, transactions, speaking styles, etc.

VoiceObjects Desktop, the graphical service creation and management environment, comes as a Web-based thin-client front-end as well as an Eclipse-based rich client plug-in to suit both administrator and developer needs. The Web front-end provides a Control Center for quick access to server and service deployment statuses, log files, trace files, system configuration, etc. The Eclipse-based development environment offers embedded features such as drag and drop, auto-completion, testing and debugging tools, automatic documentation, and much more.

Applications developed in VoiceObjects Desktop are stored in a relational database. To support moving between environments or sharing work, applications can be exported to *VoiceObjectsXML* files, an XML-based representation of call flows and dialog steps. This format is completely open and well documented. See <http://developers.voiceobjects.com/wp-content/uploads/2009/06/vo-xdk.pdf> for full documentation.

Bottom line, Voxeo VoiceObjects allows you to get your applications into a readable, standardized XML format.

In other words, no lock-in at the application layer

VoiceObjects Server, the service execution environment, is a highly scalable, flexible runtime server with features such as:

- Hot redeployment (applying changes to a production service with no downtime)
- Instant restore (backing up to a previously deployed version upon the detection of problems)
- Central cluster management (applying a change to a service with the click of a button, and making changes spread through an entire server cluster)
- Notification framework (sending SNMP traps and/or email notifications on runtime warnings or errors)

Most importantly, VoiceObjects Server dynamically renders markup code for phone applications, including VoiceXML, HTML, or custom XML, which is required for SMS, USSD, or IM integration (content markup for these channels is not yet standardized). Through its media platform driver approach, applications developed in VoiceObjects are independent of the underlying platform. Much like printer drivers, where documents look the same no matter what printer you use, phone applications will sound or look the same when deployed through VoiceObjects on any supported platform. VoiceObjects comes equipped with over 150 drivers for VoiceXML + speech recognition environments and continuously adds more drivers for new releases, new markup standards, or entirely new platforms.

Bottom line, VoiceObjects is open to any VoiceXML-enabled speech platform; it even supports the deployment of one and the same service on multiple dissimilar platforms in parallel. Of course, VoiceObjects is tightly integrated with the Voxeo Prophecy IVR platform and hosting services, which are 100% standards compliant with additional built-in functionality that prevents vendor lock-in.

In other words, no lock-in at the platform layer

VoiceObjects Analyzer, the service analysis environment, is based upon Infostore, a database for statistical data that is automatically filled by VoiceObjects Server while it processes calls. Its data model is optimized for immediate business intelligence analysis. VoiceObjects Analyzer supports the leading Business Intelligence (BI) tools: BusinessObjects, Cognos and MicroStrategy. It comes with over 60 predefined reports that are fully customizable and extendable to suit any reporting needs. The data model of Infostore itself is also extendable, open and well documented. See <http://developers.voiceobjects.com/wp-content/uploads/2009/06/vo-infostore.pdf> for full documentation.

Bottom line, VoiceObjects supports customized logging, while providing a powerful statistical data model out-of-the-box.

In other words, no lock-in at the reporting and analytics layer

For a full overview of the VoiceObjects product, please refer to <http://www.voiceobjects.com/en/products/index.html>. If you are interested in reading our fully public product documentation, please visit <http://developers.voiceobjects.com/support-training/developer-edition/voiceobjects-documentation/>.

3 Speeding migration to VoiceXML with Voxeo

In addition to being a powerful voice application environment, Voxeo VoiceObjects speeds up the task of migrating legacy applications to a VoiceXML-based framework. Note that we say speed up – there is no such thing as a magic button for automatic migration from a proprietary platform to VoiceXML... but if you do it right, you can ease the process while ensuring you don't find yourself in the same predicament in the future.

Development Productivity

First and foremost, VoiceObjects Desktop simplifies the task of creating phone applications through a variety of features that, by themselves, ease the job of creating callable, automatically documented and deployable applications. VoiceObjects is object-oriented. With this comes:

- Reusability – Re-use single objects, such as dialog steps, or entire modules, within the same application or across different projects through the use of libraries
- Inheritance – Define generic dialog behavior once and have it applied throughout the application. This includes event handling, navigation (i.e. global commands), as well as tuning parameters for ASR, TTS, or general platform tuning.

Generally speaking, VoiceObjects applications are modeled and parameterized, not coded. Examples for object types are Output, Input and Menu, where each object represents certain dialog functionality.

- Output – Presents output to the caller/user (as audio in the voice channel, video streams in the video channel or as text or images in the text and Web channels)
- Input – Comes with an embedded Output object (for re-usability!) to prompt the caller for information, with an embedded Grammar object to activate speech or DTMF grammars, and a mechanism to store user input in a Variable object or to have it directly influence runtime layer conditions
- Menu – Comes with an embedded Output object to present an introductory prompt, with menu item outputs to present each item, embedded Grammar objects per item to allow their selection, and a place to provide a target object to tell the Menu object what to do once an item has been selected

In addition to these “standard” objects, other more sophisticated object types such as Confirmation, List, or Pause provide respective dialog functionality to confirm and optionally correct a set of data items collected in the call flow, present a browsable list or even table of information, or put the system on hold during caller think times.

Each object comes with the standard functionality described above plus event handling (reacting to No Input, No Match, Help events, etc.), navigation (allowing global commands such as main menu, help, transfer to agent, etc.), and tuning parameters (to control barge-in, confidence levels, etc.). One object can constitute a fully functional, stand-alone dialog flow by itself.

When comparing the time it takes to create a callable application in plain VoiceXML to what it takes when using VoiceObjects, you can be up to 80% more productive. With VoiceObjects' Server-based features such as hot redeployment of services and central cluster management, even more time can be saved when it comes to regular change requests and service updates.

When comparing VoiceObjects Desktop to other graphical development environments, time is saved using the above-mentioned features. Developer productivity is further increased due to the incorporation of best practices that have been learned working with hundreds of customers over the past 8 years that the VoiceObjects product has been in production.

Remember, once you have developed an application in VoiceObjects, it is:

- Automatically documented in PDF and Microsoft Excel (for prompt lists) with the touch of a button
- Deployable on any VoiceXML-enabled speech platform with no further adjustment of the service
- Callable on any phone channel with adjustments only required at the presentation layer; re-use backend connections and business logic, plus the entire creation, testing and debugging environment
- Reportable out-of-the-box through leading Business Intelligence tools or your own reporting suite

We've explained how VoiceObjects can streamline development and facilitate reusability moving forward. Now let's explore what you can re-use from your existing applications when moving to VoiceObjects.

VoiceObjectsXML

The key to using VoiceObjects to migrate your existing application lies in *VoiceObjectsXML*, the XML-based representation format for call flows and dialog steps. It can be used to develop applications or initial building blocks for applications (i.e. the dialog objects), outside of VoiceObjects Desktop, in a programmatic way. *VoiceObjectsXML* code can be imported into VoiceObjects Desktop to ease the developer's job by serving as the foundation for the call flow.

As an example, consider the following code:

```
<input name="Ask for Age">
  <output type="initial">
    <outputItem bargein="false">
      <text>First, I need to know your age please.</text>
    </outputItem>
  </output>
  <output type="reprompt">
    <outputItem bargein="false">
      <text>Please say or type in your age.</text>
    </outputItem>
  </output>
  <grammar>
    <grammarItem channel="default">
      <grammarDefinition grammarType="builtin" mode="voice">
        number
      </grammarDefinition>
      <grammarDefinition grammarType="builtin" mode="dtmf">
        number
      </grammarDefinition>
    </grammarItem>
  </grammar>
  <resultHandling>
    <item object="#Age"/>
  </resultHandling>
</input>
<variable name="Age" referenceID="Age"/>
```

This shows the definition of two objects:

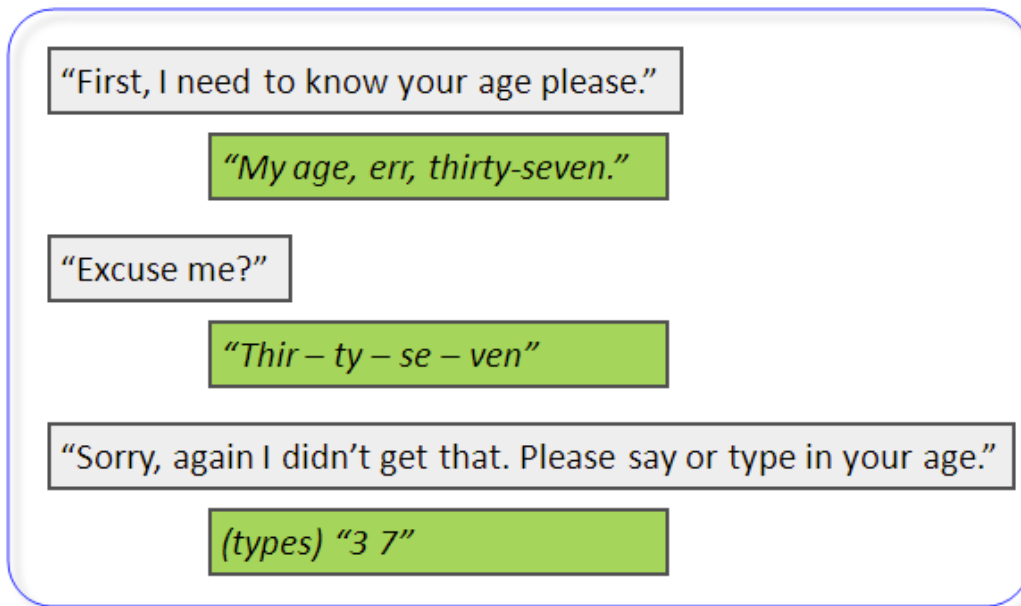
1. Input object called "Ask for Age"

- Contains an initial prompt (<output type="initial">) to say "First, I need to know your age please."
- Contains a re-prompt (<output type="reprompt">) to say "Please say or type in your age." after an event such as NoInput or NoMatch has occurred
- Provides a grammar for speech and DTMF recognition to accept numbers (<grammar>)
- Provides a reference to Variable object "Age"





2. Variable object called "Age"

- Stores the result of what the caller has said or entered
- Can be used for formatted playback in the dialog

Embedded in a Module that provides generic event handling for NoInput and NoMatch events (which does not need to be repeated in the Input object due to inheritance), a dialog that results from processing this one Input object through VoiceObjects Server could look like this:



The automatically generated project documentation export for this one object would show its entire configuration in a graphical and coherent way. Note that all properties are derived purely from the small XML snippet above.

Ask for Age			
Input Request (Initial)			
			
Language:	Default	Barge-in:	Default
Occurrence:	Always	Input mode:	Default
Channel:	Default		
Prompt:	First, I need to know your age please.		
Input Request (Event Reprompt)			
			
Language:	Default	Barge-in:	Default
Occurrence:	If >= 1	Input mode:	Default
Channel:	Default		
Prompt:	Please say or type in your age.		
Grammar			
			
Language:	Default	Weighting:	Disabled
Channel:	Default		
<i>Voice/Text Grammar</i>			
TTG	number		
Type:	Builtin		
Precedence:	Default		
<i>DTMF Grammar</i>			
TTG	number		
Type:	DTMF		
Precedence:	Default		
Result Handling			
Variable:	 Age	Slot:	(empty)
Options			
Mask caller input for System DB logging			False
Grammar control:			Default
Increased dialog timeout:			Default
Enable auto-advance on No Input			False
Enable auto-advance on No Match			False
Record utterances:			Default
Recording scope:			Default

In a nutshell: once you have the *VoiceObjectsXML* representation of your dialog, you have an immediately callable and documented application. And it doesn't take much more to enable this application for SMS, USSD, IM, or the mobile Web.

So how do you arrive at the *VoiceObjectsXML* code? Basically by applying a script that converts any kind of base material you have in your project to this format. Let's have a look at different types of code or documentation that you can typically re-use for migration.

Consider the following sample prompt list from an Excel spreadsheet:

	A	B	C	D	E
	Dialog	Name Audiofile	Notes	Reference Visio	Prompt Content
1					
2	MEN_Welcome				
3		Welcome	[DTMF = Voice]	Page 1, Prompt 1	Welcome to Customer Self-Service.
4	MEN_Main				
5		MainMenu	[DTMF = Voice]	Page 1, Prompt 2	This is the main menu, please select one of the following options.
6		MainMenuReenter	[DTMF = Voice]		Back in the main menu.
7		ServicePlans	[DTMF = Voice]		Service plan manager.
8		CustomerData	[DTMF = Voice]		Customer data.
9		Billing	[DTMF = Voice]		Billing.
10		Support	[DTMF = Voice]		Support.
11		MainMenu2ndNi	[DTMF = Voice]		I still did not hear you. Please say service plans, customer data, billing, or support.
12		MainMenu2ndNm	[DTMF = Voice]		I still did not get that. Please say service plans, customer data, billing, or support.
13	CC_Type				
14		CreditCardType	[DTMF = Voice]	Page 2, Prompt 1	First, I need your credit card type. Please say Visa, American Express, or Mastercard.
15		CreditCardTypeReenter	[DTMF = Voice]		Again I need your credit card type. Please say Visa, American Express, or Mastercard.
16		CreditCardType2ndNi	[DTMF = Voice]		I still did not hear you. Please say Visa, American Express, or Mastercard.
17		CreditCardType2ndNm	[DTMF = Voice]		I still did not understand. Please say Visa, American Express, or Mastercard.
18	CC_Number				
19		CreditCardNumber	[DTMF = Voice]	Page 2, Prompt 2	Please tell me your 16-digit credit card number.
20		CreditCardNumberReenter	[DTMF = Voice]		Again I need your 16-digit credit card number, digit by digit.
21		CreditCardNumber2ndNi	[DTMF = Voice]		I still did not hear you. Please say or type in your 16-digit credit card number.
22		CreditCardNumber2ndNm	[DTMF = Voice]		I still did not understand. Please say or type in your 16-digit credit card number.

Look closer at the documentation above. It includes almost everything you would need for functional dialog objects in a VoiceObjects application:

- Audio file names, which can also be used as Audio object names
- Dialog step names, which can be used as Input/Menu/Output object names
- Prompt texts
- Naming conventions (such as “Reenter” or “2ndNi”) that can be exploited to define correct event handling

You might even want to include the notes and Visio references in the description fields of the objects to be created. This way you don’t even lose the documentation references.

All it takes now is writing a VBA script that iterates over all lines, extracts the respective data from the cells, and produces *VoiceObjectsXML* code. The same approach works with Word-based project documentation. VBA scripts can iterate through all pages and identify the parts

where prompt information is coded. If you spend some more effort identifying call flow logic, you can achieve even higher reusability rates.

In the end, the generated *VoiceObjectsXML* code gets imported into VoiceObjects Desktop, allowing you to kick-off your implementation with a multitude of pre-configured objects that are ready to be used in your call flow. This is a proven approach that has saved companies with large migration projects hundreds of hours.

Application code

Another source of reusability is the code that represents your existing dialogs. If you currently use a development environment that does not come with any form of readable, text-based (XML or proprietary) code for your applications, then the only viable migration approach for you would be the one explained above, based on documentation. If you do use an IDE that allows an export of applications or already stores call flows in a readable format, the code should contain information that you can use to identify reusable components. By writing a conversion script that takes the code and parses it to find prompt information or any other kind of code that could help you prepare dialog objects, you can produce *VoiceObjectsXML*. As an example, the Storyboard Manager, an Excel-based prompt management tool integrated into VoiceObjects Desktop, already comes with such a script to read **Nortel PeriPro** project files. It is a simple script that extracts all prompt-related information from those files and imports them into a spreadsheet form. As the Storyboard Manager comes equipped with other macros that generate *VoiceObjectsXML*, you will save a lot of time by not having to manually create the basic building blocks for your application.

While this paper mainly talks about migration to VoiceXML, you might be in a situation in which your applications already are in VoiceXML, but not in an environment like VoiceObjects. Through *VoiceObjectsXML*, you can even leverage plain VoiceXML to get your call flows into the Voxeo VoiceObjects platform. VoiceObjects also allows you to maintain the main menu or other main modules of your services within its environment, while branching out to existing, plain VoiceXML-based applications at call time if necessary. This helps with step-by-step migration, instead of waiting for the whole set of applications to be converted before you can exploit the great features of VoiceObjects.

Other Resources

Obviously, getting the call flow migrated is only part of the job. A voice application can include audio files, grammar files, external script code, backend connectors, CTI integration, etc. The good news is that ALL of this can be reused with VoiceObjects. Due to its nature of being a runtime application server, it is basically agnostic towards any external resources like the ones mentioned above. It is up to the underlying media platform you use whether you can reuse those files as is, or whether you need to convert them. For conversion of audio files,

use any publicly available conversion tools. For conversion of grammar files, use NuGram IDE, an Eclipse plug-in for grammar management bundled with VoiceObjects.

Here are some examples of application components and how to reuse them:

Audio files

- Place them on a resource server or just keep them where they are today. Simply configure the URLs in VoiceObjects through generic Resource Locator objects.

Grammar files

- Place them on a resource server or keep them where they are today. Configure the URLs in VoiceObjects through generic Resource Locator objects.

Script code

- Place them on a resource server or keep them where they are today. Configure the URLs in VoiceObjects through generic Resource Locator objects.
- VoiceObjects ships with an ECMAScript engine. If this is not the programming language you are used to or using in your current environment, VoiceObjects offers a flexible Script Bus architecture to allow plugging in any script engine you need.

Backend connectors

- VoiceObjects provides four types of connectors: Java, CGI/HTTP, Web Services, and database. If you already use Java or server-based CGI scripts/Web services to access backend systems, you are good to go. If not, add a Java wrapper around it. If you have a relational database system in use, simply communicate with it over a JDBC connection.

Voxeo's own Prophecy IVR includes technology that automatically translates speech grammars from one format to another on the fly – without any work at all required from the developer. Prophecy supported grammar formats include SRGS-XML, SRGS-ABNF, Nuance GSL, IBM BNF, JSGF with SRGS SISR, Speechworks SISR, and Nuance semantic interpretation. This capability eliminates the cost and time previously required to change ASR engine vendors or versions.

4 Migration lessons learned

We want to be honest. There is no magic bullet for migration away from your current infrastructure. But if you are facing platform end-of-life or working with a sub-par VoiceXML platform, the investment will pay off. In this whitepaper we summarize what we believe is a worthwhile effort even if you do not immediately face end-of-life of your current platform – and why it is a good idea to choose Voxeo.

Using VoiceObjects will result in more stable, feature-rich applications and save you time in the mid- to long-term with automatic documentation, automatic reporting, centralized cluster

deployment, and many more features. *VoiceObjectsXML* is an open, flexible format for call flow representation that is ready to be exploited through automatic conversion scripts to help you simplify migration and save money.

5 Voxeo Unlocked Communications

Now that we've discussed the VoiceObjects and application migration, let's back up because choosing the right service creation environment is only one piece of the overall solution. You still need a platform for your application. It's critical to recognize that many platform providers claim openness, but use proprietary development tools and tags to maintain ownership of your application. As mentioned previously, Voxeo VoiceObjects is a platform-independent solution that enables one-click deployment to virtually every modern IVR platform offered today. That said, Voxeo's own Prophecy platform and Prophecy Hosting services offer compelling advantages over other IVR platforms:

Standards Compliance: Voxeo's architecture is based entirely on open standards such as VoiceXML 2.0 and 2.1, CCXML, SIP, MRCP and SSML. Voxeo offers the only solution that is 100% VoiceXML compliant, having passed every mandatory and optional certification test.

You own your application: With Voxeo, you own your intellectual property, resulting in the flexibility to switch providers, bring your application in house, or engage your choice of development resources.

Compatibility: In addition to the functionality offered by VoiceObjects, the Voxeo Prophecy platform itself includes compatibility modes that make porting VoiceXML IVR applications as easy as changing a single Prophecy setting.

Third-party development tools: Voxeo Prophecy and Prophecy Hosting give you the flexibility to use VoiceObjects or any VoiceXML compliant third-party development tool.

Multi-channel: In the same way that VoiceObjects supports "develop once, deploy anywhere" functionality for multi-channel phone applications, the Voxeo platform meets your Unified Self-Service needs with existing support for voice, SMS, IM, Twitter, and mobile web communications.

Speech flexibility: The Voxeo platform includes built-in speech recognition and speech synthesis (ASR and TTS) engines, and supports any MRCP-compliant third-party speech technology.

Deployment flexibility: Unlike other vendors, Voxeo offers both on-demand and on-premise deployment options with the ability to easily move from one to the other or leverage a hybrid model. For instance, some customers deploy a solution on-premise and use Voxeo hosting

for overflow and disaster recovery. Voxeo's on-premise Prophecy platform is the very same product used to power Voxeo Prophecy Hosting.

Telephony options: Voxeo's IVR platform is natively SIP-based. Customers can connect to the platform either directly through VoIP/SIP or via traditional TDM circuits via VoIP gateways. Voxeo owns its SIP stack and can ensure compatibility with any SIP-based environment as needed.

6 Learn More

Get in touch with us today to find out how you can leverage your existing source code and documentation for faster and easier VoiceXML migration. Better yet, find out how you can unlock the potential of your communications applications with a flexible, robust, and standards-based solution. At Voxeo, we choose to keep our customers by providing a better product and better support – never by locking them in. Contact us today!



VOXEO

sales@voxeo.com

USA +1 407 418 1800

UK +44 20 7887 6085

Germany +49 2204 845 100

China +86 10 8282 5011