



# Security, Audit Trails, Team Collaboration, Version Management

Dr. Andreas Volmer,  
Presales Manager EMEA  
VoiceObjects GmbH  
January 2009

## **TABLE OF CONTENTS**

---

<b>MANAGEMENT SUMMARY</b> .....	<b>2</b>
<b>SECURITY</b> .....	<b>2</b>
USER MANAGEMENT .....	2
PASSWORD POLICIES .....	3
AUDIT TRAILS .....	3
<b>TEAM COLLABORATION</b> .....	<b>4</b>
OBJECT LOCKS .....	4
OBJECT METADATA .....	5
<b>VERSION MANAGEMENT</b> .....	<b>6</b>
VERSION MANAGEMENT IN THE METADATA REPOSITORY .....	6
INTEGRATION WITH AN EXTERNAL VERSION MANAGEMENT SYSTEM .....	7
<b>CONCLUSION</b> .....	<b>8</b>
<b>FURTHER READING</b> .....	<b>8</b>



## Management Summary

The VoiceObjects platform allows for collaborative development of phone services in environments that are highly sensitive to security and audit requirements. In order to meet these requirements, VoiceObjects features built-in, role-based user management with granular access control lists, project and version management, object locking mechanisms, and configurable audit trails.

All of this also works in multi-tenant environments where each tenant can manage their own users. The VoiceObjects platform's built-in security mechanisms ensure that no tenant will ever see another tenant's projects, services, log or trace files.

This white paper gives an overview about the relevant features offered by the VoiceObjects platform and suggests best practices for defining processes.

## Security

Access control and grants policy management is based on VoiceObjects User Management, which also covers securing operations and management related activities.

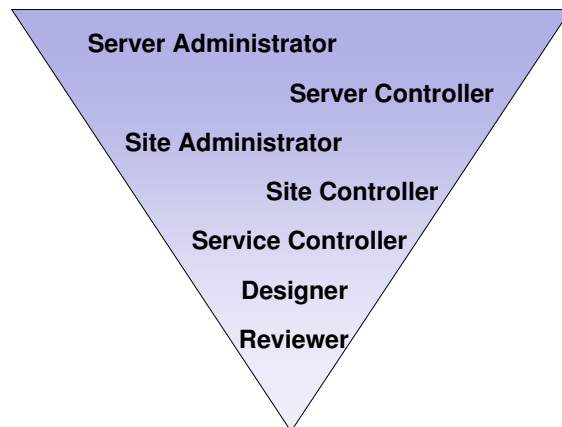
### *User Management*

VoiceObjects comes with role-based user management. Every VoiceObjects user has a specific role, like *Reviewer*, *Designer*, *Service Controller*, *Observer*, *User Manager*, or *Server Administrator*, defining the functionality available to him.

In addition to that, the VoiceObjects administrator maintains access control lists (ACL) defining which servers, services, projects and libraries each individual user he has access to.

Using this approach, for example, a setup can be configured where a project manager has full control over an application project, while individual team members can only work in project libraries that are referenced by the central project. This allows for granular control over which objects and modules each individual developer may modify.

Access rights are consistently and centrally checked for any user request through VoiceObjects Desktop for Web, Desktop for Eclipse, or through the Web Service interface. Any user request that was either denied or granted by the system – be it a service redeployment, the shutdown of a server instance, or just the modification of an object in a project – can be logged in audit trails (see section on Audit Trails below).





## **Password Policies**

VoiceObjects supports password policies covering the format of allowed passwords, rules on how and when to change passwords, and security mechanism to prevent unauthorized login attempts.

To ensure that user-defined passwords meet given security standards, password patterns can be defined that need to be met by new or changed passwords.

A non-exhaustive list of configurable password requirements includes:

- Password must have a certain length;
- Password must have at least one digit;
- Password must contain at least one non-alphanumeric character;
- Password must contain both upper and lower case characters;
- Password must not be derived from user name or user ID.

On top of that, VoiceObjects can be optionally configured such that

- Initial passwords are automatically assigned on user account creation and sent to the user per email;
- Users are required to change their password on first login;
- Passwords expire after a  $n$  days;
- New passwords must not be equal to any of the  $n$  preceding passwords.

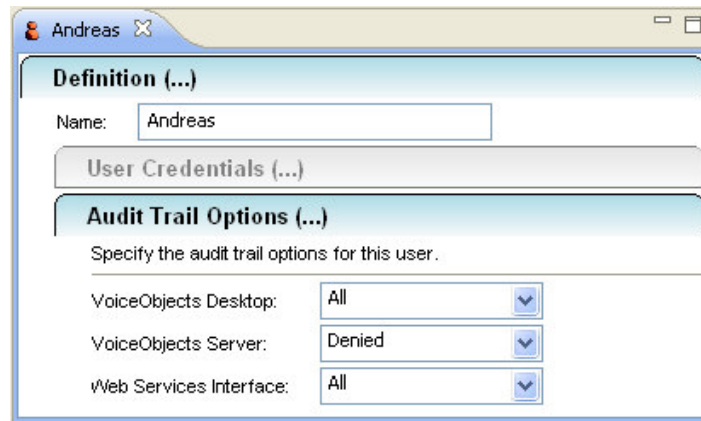
In environments with high security concerns it is advisable to separate the management of users from the management of the installation. In this case, the appropriate user roles are *User Managers* on one side and *Server Controllers* or *Site Controllers* on the other side. In this scenario, user accounts are created and maintained by *User Managers*. This is their only task, and a modification made by one *User Manager* needs to be crosschecked and validated by a second *User Manager*. This prevents single *User Managers* from expanding their capabilities by creating users of other types (e.g. *Designers* or *Server Controllers*).

## **Audit Trails**

Auditing is typically used to track activity in production deployments, or to detect manipulation attempts. It can also be used in a development environment to track development activities. In VoiceObjects, this includes tracking the creation, modification and deletion of objects, as well as activities on services, servers, and successful and failed login attempts.

The VoiceObjects Server Administrator can adjust the level of detail of audit trails to your security experts' requirements. By default, only activities such as failed login attempts or denied access to Control Center functions are logged.

This screenshot shows how Audit Trail Options are configured for user "Andreas":



For any activities within VoiceObjects Desktop, on VoiceObjects Server, or through the Web Service Interface, the following audit trail options can be defined:

- None:** No audit data is written.
- Denied:** Only user activities that were denied due to insufficient privileges are written to the audit log files.
- All:** All user activities are written to the audit log files.

When activating the Audit Trail Option “All” for VoiceObjects Desktop and Web Services Interface, you will be able to trace, among others, information on

- Timestamp of the activity;
- IP address from which the request was made;
- User ID of the user who made the request (and his role);
- Site ID (in a multi-tenant setup, this specifies the tenant the user belongs to);
- a message describing the activity, like which object was created, modified, or deleted;
- whether the request was granted or denied.

## Team Collaboration

VoiceObjects fully supports teams collaborating in the development of automated phone services. Several team members can work on the same project at the same time, using either Desktop for Web or Desktop for Eclipse – each developer can use his preferred development environment.


### ***Object Locks***

VoiceObjects developers use the object library supported by VoiceObjects Server to create phone applications. They organize these objects in call flows and configure them according to the call flow requirements. This call flow and object configuration is

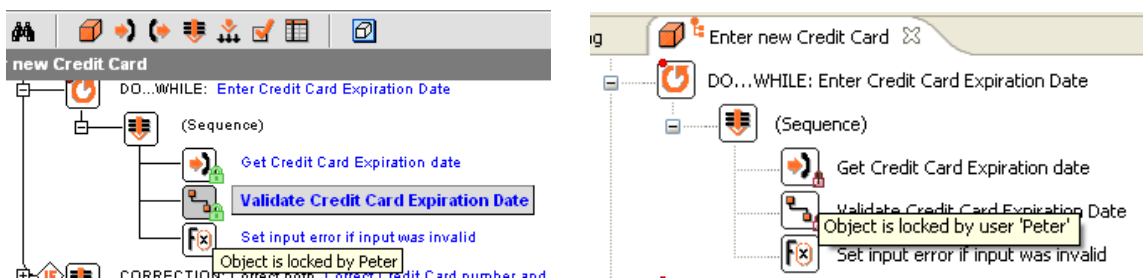


persisted in VoiceObjects' Metadata Repository, a database hosted by a standard database server such as Oracle or SQL Server.

Since all developers are working on the same central application definition, no “merging” activity is necessary to consolidate the team’s efforts. In order to prevent developers from overwriting each other’s changes, they can lock individual objects.

Locked objects are marked with a little padlock symbol , both in the Object Browser and in the Dialog Designer. For the user who locked the object, the padlock is green; for all others, it is red. There is also a tooltip associated with the padlock indicating the user who locked this object.

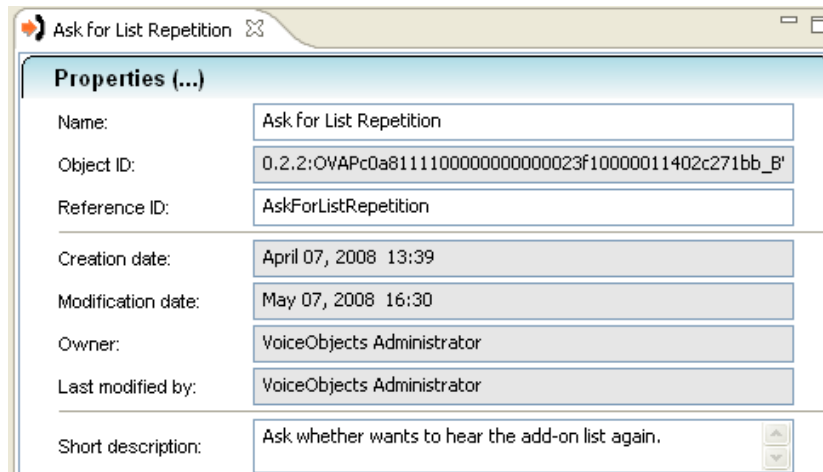
The following two screenshots show two versions of Dialog Designer: On the left, we are looking at Peter’s Dialog Designer in Desktop for Web. He has locked two objects. On the right, we are looking at Debbie’s Dialog Designer in Desktop for Eclipse. She is working on the same project as Peter and sees that Peter locked two objects. She can still open these objects and review their configuration, but she cannot change and save them – until either Peter or a Server Administrator unlocks them.



## Object Metadata

For each object that was created by some developer, VoiceObjects maintains metadata. This comprises a unique reference ID, a description and comment that can (and should!) be added by the developer.

In addition, VoiceObjects keeps track of who created the object and who applied the most recent modifications – and when these activities happened. This metadata is available both in Desktop for Web and in Desktop for Eclipse. The following screenshot shows this information in an object editor within Desktop for Eclipse. Note that the grey text fields cannot be changed by the developer; they are automatically maintained by VoiceObjects Desktop.



This object metadata is also available as search criteria when searching for objects within VoiceObjects Desktop. It is possible, for example, to search for objects

- that were created after a certain date;
- that were modified after a certain date;
- that were locked by yourself or by somebody else.

## Version Management

In most VoiceObjects projects, two mechanisms of version management are being used:

1. The built-in project version management within the Metadata Repository database;
2. Integration with an external, file-based version management system such as CVS.

Both approaches, which are used in combination in most projects, are described below.

### ***Version Management in the Metadata Repository***

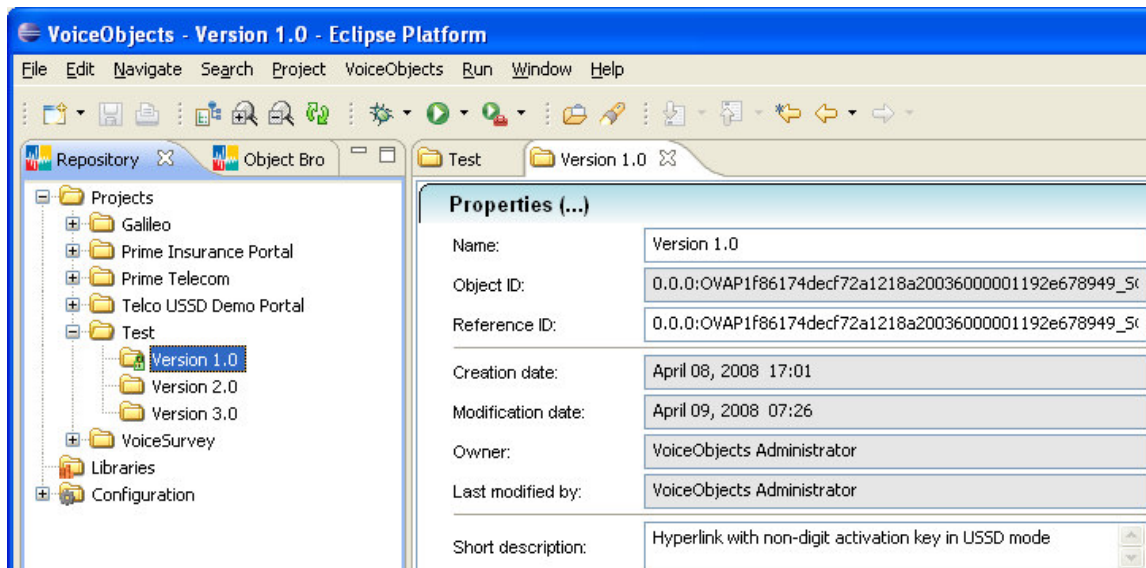
For each project within the Metadata Repository database, an unlimited number of versions can be maintained. Each version contains a snapshot of the entire project. Versions can be locked so that they cannot be modified any longer (as long as the lock is applied); it is also possible to have several unlocked versions of a project representing different branches of a project.

It is recommended to create (“publish”) a new project version for every project status that is deployed to the production system. That way, analysis results comparing different deployment versions with regard to customer behavior or recognition success can always be traced back to the respective application versions. This process also ensures that you can roll back to any relevant historic project version at any point in time.

The following screenshot shows the project browser within VoiceObjects Desktop for Eclipse. The VoiceObjects developer has opened “Version 1.0” of the project named “Test”. For both the project and for all of its versions, VoiceObjects automatically



maintains metadata such as creation and modification date and according user information.



In environments where both the development and the production system use the same database server for the Metadata Repository, it is recommended to use project versions to control the staging process. One version will represent the “tip” revision currently modified by the development team, while another (locked) version will represent the most recent “stable” version that was signed off by the quality assurance team and that is currently in production.

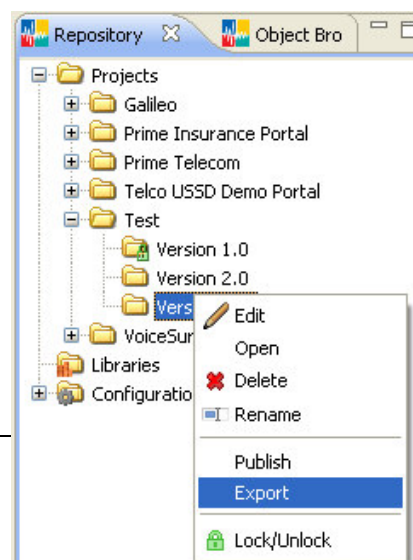
### ***Integration with an external Version Management System***

In addition to the built-in, database-centric version management, most development processes also require the integration with an external, file-based version management system such as CVS.

This makes sense because phone application projects not only consist of the call flow logic (which is represented by the VoiceObjects project), but also need to maintain a large number of external resources such as audio files, grammar files and traditional source code implementing back-end access.

VoiceObjects allows developers to export project versions to VoiceObjectsXML, a markup format defined by VoiceObjects. A VoiceObjectsXML document is a 1:1 representation of the project definition in the Metadata Repository. It can be re-imported through VoiceObjects Desktop at any time in the future, allowing for roll-back to earlier versions.

Since VoiceObjectsXML is a file-based format, it can be version controlled by any file-based version management system.





Note that not only entire project versions can be exported and imported to or from VoiceObjectsXML. You can also export or import single objects or modules. This allows for merging developers' work that was created when not connected to the central Metadata Repository database.

When working with CVS or other file-based version management systems, developers typically use Eclipse plugins to facilitate the process of checking in/out resources. With VoiceObjects Desktop for Eclipse – which, to its core, is just another Eclipse plugin – VoiceObjects developers hence don't have to leave their development environment for version control of VoiceObjects projects.

When using the built-in VoiceObjects version management in combination with an external version control system, it is best practice to synchronize the version numbers or tags in both systems.

## Conclusion

Based on its database- and server-centric architecture, many features in VoiceObjects ease the development of phone applications in teams of developers striving for high quality and efficient implementations.

Built in user management and access control, version control, change management, auditing, and object locking ensure that developers can work without disrupting each other's work on large projects. The same features ensure tight security in production environments, even in multi-tenant environments.

## Further Reading

In the VoiceObjects Online Help, the following chapters supply more detailed information on the topics discussed in this paper:

### Administration Guide:

- *User Management – Basic Topics*: Managing Users, user roles and access control, audit trails
- *User Management – Managing Sites*: Setting up a multi-tenant environment

### Desktop for Eclipse Guide

- *Repository Browser*: Project and version management
- *Basic Commands / Lock or Unlock Objects*: Objects locking