

## Servoy Comparative Functionality Guide for Microsoft® .NET

---

### Executive Summary

Traditionally, ISVs (Independent Software Vendors) have been very successful developing and delivering their products with so-called 4GL environments such as: Uniface, FoxPro, Progress, Access, PowerBuilder, Magic, Filemaker Pro, etc. However, these 4GL environments have not kept up with modern technology trends with limited or no support for web applications, Software as a Service (SaaS), or multi-site deployment; and these environments use proprietary, vendor locked-in technologies which isolate them in corporate environments.

The solution for ISVs is to move to more open standards-based and pervasive environments such as Java and .NET. Servoy is a Java-based development and deployment environment that makes programming in Java much more productive – solving the traditionally difficult deployment issues and the steep learning curve of Java.

When considering Servoy and Microsoft's .NET platform, ISVs have to take into consideration the following issues:

*1. Is time to market an important issue for me?*

Independent studies show that Servoy allows projects to be completed 2-5 times faster than using .NET.

*2. Is it important to offer a cross-platform application?*

Microsoft .NET applications only run on Microsoft's own operating system. While Windows is currently the most dominant operating system, Linux, Mac OS and Solaris adoption are growing quickly. Without any additional development effort, Servoy can be deployed to all modern operating systems at the development, server and client levels.

3. *Do you often revise your software?*

When you release new versions of your software, Servoy enables you to roll forward and backward between releases. Because of Servoy's modular code structure, your customers have the ability to customize Servoy applications without breaking the "base" functionality.

4. *What is the skill set of your development team?*

Servoy can be learned in a very short time – the basics can be learned in a week. As part of Servoy's ISV Assurance Program, employees are re-trained on the job with guaranteed results – so there is no need to invest in expensive external training which often can lead to unqualified and unexpected results.

5. *Do I need to support multiple or mixed deployment scenarios?*

Servoy applications can be deployed as client/server; over the web; stand-alone and to mobile devices – all from a single code base. In applications with a .NET user interface, many times core business logic needs to be re-written for each target deployment platform.

6. *Does my product compete with any of Microsoft's offerings?*

Microsoft offers CRM; ERP; Accounting; Project Management; POS and other business software solutions. If you, as an ISV, offer solutions in any of these areas, you will have to consider whether you want to build your solution on a platform that is supplied to you by your competitor. Servoy does not compete with ISVs – rather we partner with them.

7. *Does my vendor offer a program that specifically helps ISV's?*

Servoy has a proven program in place – the Servoy ISV Assurance Program – that helps ISVs migrate from their current environment to Servoy in a fixed time frame for a fixed price with a flexible payment schedule, including all development and end-user licensing fees.

8. *Is my vendor committed to my success?*

At Servoy, we can't succeed unless you do. We work with our ISV customers as partners in their success. Our sales and development teams work closely with your staff to help you every

step of the way. We listen to our customers, often incorporating specific features into the Servoy Product Suite at their request. We are also accessible. Servoy is big enough to give you the support you need and small enough to give you direct access to our C-level executives on an as-needed basis. If you are an ISV executive, you are welcome to contact our CEO, Jan Aleman directly by email at: [jaleman@servoy.com](mailto:jaleman@servoy.com); or by phone at: +31 33 455 9877.

In comparing and contrasting the benefits and limitations of Servoy and .NET, it's important to understand the who, what, and how of each environment.

## About This Whitepaper

The purpose of this whitepaper is to help ISV executives and developers understand the technology differences between Servoy and .NET in terms of architecture, development methodologies and integration techniques; and to show a real-world TCO (Total Cost of Ownership) example.

Servoy and .NET are completely different products – each with its own set of strengths and limitations. The key is to choose the right tool for the right job.

## Who is each environment targeted at?

### *.NET*

.NET was first introduced in 2002 as an alternative to the Java platform. .NET is used by medium to large workgroups within corporations and some ISVs – both of whom do not need a cross-platform solution; intend to use “Windows only” technology; and already have a heavy investment in other Microsoft technologies, such as SQL Server and ASP.

### *Servoy*

Servoy, first introduced in 2001, is built on the Java platform designed by Sun (then a major Microsoft competitor) and supported by large software companies such as: Oracle, IBM, Apple, etc. Servoy is used by ISVs, medium to large workgroups within corporations, and independent software consultants who build custom applications for their customers.

## What are each environment's core competencies?

### *.NET*

The .NET framework allows developers to write applications in several different languages (C#, VB.NET, J#, ASP.NET); and offers tight integration with other Microsoft technologies like SQL Server.

.NET applications can be deployed to Windows clients via a native application or as a web application using ASP.NET. The server must be a Windows machine. The client must be a Windows user by default as well. Although it is possible to write browser applications that run cross-platform, this is not the default in .NET and requires significantly more effort.

The .NET framework provides various class libraries, which are used by the developers to create their own applications. The applications then run in the Common Language Runtime (CLR), which acts as the application virtual machine. The class libraries and the CLR together compose the .NET framework. The entire .NET framework is managed by Microsoft and is completely closed-source.

### *Servoy*

Servoy has the ability to deploy virtually the same application via a cross-platform native client – Servoy Smart Client – via Java WebStart on Mac OS X, Windows, Linux, and Solaris; and via a web browser – Servoy Web Client – on any platform that supports JavaScript including: Mac OS X, Windows, Linux, Solaris, Windows Mobile Devices, iPhone, Palm Treo, and others. The same code base can be used to deploy as a Smart or Web Client, without the need to rewrite the application. Both business rules (application logic) and user interfaces can be shared across platforms and deployment methodologies.

Servoy comes bundled with iAnywhere's SQL Anywhere, an enterprise-level SQL database at no additional charge – but ANY (or multiple) SQL databases can be used including: Sybase, Oracle, IBM DB2, MS SQL Server, MySQL, PostgreSQL, or any other ANSI 92-compliant SQL database with a JDBC driver.

Servoy Server can run on any operating system platform including: Mac OS X, Windows, Linux, Unix, BSD, or Solaris; and will scale based on hardware – from laptops to IBM Z Series mainframe computers.

Java has been around for eight years longer than .NET and Sun recently released Java under the GPL license, making it open-source and truly free.

Developers write their applications inside Servoy using JavaScript, or with pure Java. Servoy offers a JavaScript layer for common business logic and workflow. At deployment time, this JavaScript is compiled into native byte code for performance and security. You get the ease-of-use of a scripting language, combined with the security and performance of a compiled language. In addition, Servoy comes with hundreds of built-in functions that allow you to seamlessly implement functionality without having to write all the code yourself – enabling your application to have fewer lines of code.

## How do these products work from an overall architectural point of view?

.NET and Servoy take nearly opposite directions when it comes to overall architecture. .NET is based on proprietary technology; and Servoy is based on 100% open-source Java. Below, we will compare and contrast the two environments to give you a basic overview of the product lines and what each product is used for.

### *.NET*

.NET offers a client/server technology by deploying web-based applications written in ASP.NET on a Windows Server running IIS. For rich client desktop applications, .NET doesn't provide any tools for automated deployment of your application. The developer must compile the application and handle the install on each client. There also isn't an application server for .NET. – meaning there is no centralized way to manage your solution and clients.

Most .NET developers use the Visual Studio IDE (Interface Design Environment) to develop their applications. There is no built-in version control with .NET, although you can setup your own SVN/ CVS server to manage sharing code with multiple developers.

Other Microsoft technologies, like SQL Server and Active Directory, have API's available for .NET so developers can write code to integrate those services into their applications. Note that while an API is available, the .NET framework doesn't do much for you automatically. The .NET developer must control everything manually in their code.

## *Servoy*

Servoy is based on Java client/server technology and includes Servoy Developer – an application builder; Servoy Smart Client – a rich desktop Java-based native client; Servoy Web Client – a 100% HTML/CSS/AJAX version for web browsers; and Servoy Server – an application server. Servoy also offers an optional “Multi-Developer Server” that allows more than one developer to work on the same project at the same time – and broadcasts all changes to all members of the development team; Servoy Headless Client – a headless Java client API that allows developers to write JSP pages to interact with their applications and with other web services; – and included at no extra charge with the standard Servoy Server; and Servoy Disconnected Client – that runs on a disconnected computer, and enables synchronization back to one or more SQL sources using the optional MobiLink software from iAnywhere.

Servoy interacts with a backend database (or multiple databases) over JDBC, and can handle all of the SQL queries for you. Writing your own SQL is available, but not required. With Servoy, you can integrate with other services, such as: LDAP, SOAP, etc. through the Servoy Java API; and there are many bundled Servoy plugins that have already been written to interact with these technologies. External components such as Java Applets, JavaBeans and other Java classes are also zero deployment; and can be automatically pushed to remote servers and to native client users without needing additional software.

## Looking at the “How” of Building Applications

### *Building the .NET Application*

Developers usually build their applications with Visual Studio. The application(s) consist of physical disk files on the developer's local machine. Managing the database server is done outside of the application, usually with an administrator tool provided by the database vendor.

Desktop rich client applications are usually written in C#, VB.NET, or J#. When using the Visual Studio IDE, you can use the compiler and GUI (Graphic User Interface) builder to develop your applications. The IDE also provides a way to connect the GUI to your code, but all of the data-binding must be handled manually in code by the developer. The same thing must be done for data-broadcasting, since the .NET framework does not provide broadcasting events and data between clients.

Web-based applications are usually written in ASP.NET. All of the GUI must be done in HTML. Any GUI that was originally built for a rich client application cannot be reused, but must be completely recreated in HTML. However, the .NET framework does provide a way to share business logic between a web-based, and desktop-based application.

### *Building the Servoy Application*

Developers build Servoy applications using Servoy Developer and deploy them to Servoy Server. Users can then connect to the Servoy application server via Servoy Smart Client or Servoy Web Client to interact with their application.

Servoy has *no proprietary file format*. All the information about your solution is stored as metadata in the form of rows and records in a SQL database repository (the repository can be any SQL database you choose).

Servoy Developer is a visual IDE and comes complete with a wide range of built-in functions and plugins which are used by the developer to create their own application. The entire Servoy Developer application runs in the Java Runtime Environment (JRE).

When building both desktop rich client and web-based applications, you can use the exact same business logic and the same Servoy Developer solution to seamlessly deploy on both environments. You don't need to recreate any GUI or business logic.

Servoy developers can also take advantage of a number of form and data elements – from “portals” that show related data; to buttons; background graphics as buttons; as well as line, box, circle, and polygon drawing tools. Developers can also specify auto-enter data; and specify columns to

contain data looked up from other tables – handled automatically by Servoy.

Servoy also supports JavaBeans and Java Applets directly within the application. A JavaBean is a reusable software component that can be visually manipulated in builder tools. A Java Applet is a small Java application that usually runs inside a browser. JavaBeans usually have some kind of user interface element to them such as: a slider control, a clock, a drag-and-drop pane, etc. A Java Applet is generally a small application that does something: has a monetary conversion, displays molecule data in 3D, etc.

In addition, the Servoy Plugin API enables developers to easily create their own custom plugins, or use third-party Java classes already available – so that Servoy applications can interact with Web Services, SOAP, LDAP, etc.

Servoy also supports events on objects. For example, field events include: onFocusGained, onFocusLost, onDataChange, and onAction. Form-based events include: onShow, onLoad, onRecordEditStart, onRecordEditEnd, onRecordSelection, onHide, as well as including override events for all standard applications menu options: onFind, onNextRecordCmd, onPreviousRecordCmd, onNewRecord, onDeleteRecord, etc. Having such a wide range of events at the Servoy developer's disposal is a huge help when it comes to enforcing business rules; re-using code across an application; dynamically changing or highlighting data; showing/hiding/graying-out objects, etc. Support for these events, provides Servoy developers with the control they need to create adaptable, configurable, flexible applications.

Overwhelmingly, Servoy customers have reported that their developers are able to create programs 5-10 times faster using Servoy than other development environments, like .NET.

## Deploying the Completed Applications - Client/Server

### *Deploying a .NET Application*

The .NET framework must be installed on each client in order to run any desktop version of an application. In order to install the framework, the user must have Administrator privileges. For Windows XP and prior versions, the .NET framework doesn't come installed by default. However, it is installed on all versions of Vista. New versions of the .NET framework introduce

new functionality and in many cases are mutually exclusive. For example: Applications developed on .NET 1.2 might not run on 2.0 and vice versa. Larger corporations that have applications developed in both versions are already starting to experience this restriction as a serious problem.

Using Visual Studio, you must compile and export your solution as a Windows Installer file. From there, it is up to you to handle installing the application on each client. You must either build your own deployment solution, or purchase a third-party solution (if available). The same scenario applies to other areas like data broadcasting, versioning, and client management.

The .NET developer must also handle solution versioning, and client updating. .NET doesn't provide any of this functionality built-in – resulting in a huge problem with large sites and applications that need to be updated often. If an application has been installed to 1000 desktops and a new version is rolled out, this new version has to be installed simultaneously on all desktops for correct upgrading. Although there are tools to semi-automate the installation, it is a very error-prone process.

Usually, WAN (Wide Area Network) clients must access a rich client .NET application through Terminal Services (TS) for performance. This additional software layer causes many configuration headaches, including: printing; printer setup; access to local files; access to local hardware; sharing data with applications running locally; etc.

### *Deploying a Servoy Application*

The Servoy Client software also has to be installed on each client computer – but rather than installing manually or performing a network install – the client has only to open a browser (pointed to the IP address of Servoy Server) and click one button. The client software will download itself one time (about a 3MB download); install itself; configure itself; and then continue to launch. No administration rights are necessary for this install and the installed client runs in an isolated, secure sandboxed environment. Both the client and the developed application are cached locally.

Then every subsequent time the Servoy application is launched – from a web browser; a desktop shortcut; or via a “deeplink” on your existing website or portal – it checks with the Servoy application server to see if an update is needed. If there is an update, Servoy will download, install

and configure itself automatically and then continue to launch. You NEVER have to manually update client software – ever – with Servvoy.

Once the Servvoy developer – or development team – has their application ready for deployment, it's a simple matter of exporting the solution out of the Servvoy repository database (using Servvoy Developer), uploading it to the application server (via a web page) and telling the newly installed version to activate by checking one checkbox. You can roll forward to any new release, or roll back to any previous release at any time. There is no limitation on the number of releases you can have – PER solution, PER server.

## Deploying the Completed Applications – Web

### *Deploying a .NET Web Application*

Most web applications are written in ASP.NET, and can be deployed by uploading the .asp files to the Microsoft Web Server running IIS. All of the GUI is done in HTML, and the business logic is mostly ASP.NET with the possibility of integrating with other .NET languages.

Microsoft's Visual Studio encourages programmers to code user interfaces that only run on the company's web-browser IE 6 for older versions of .NET and IE 7 for newer versions of .NET. There are some major differences in rendering between these two versions of IE, causing the necessary issues of supporting both browser versions.

The web server is not included in the .NET framework, but must be setup and maintained separately by the developer. The only supported webserver by default is IIS, not the first choice today in web server technology from a security, scalability and availability standpoint.

Since clients are accessing the .NET application through a web browser, the rendering of the GUI and printing will be based on their browser and browser-related settings – causing inconsistency and headaches when needing precise, standard printed forms or consistency between multiple browsers.

## *Deploying a Servvoy Web Application*

Servvoy offers two different ways to deploy your application to the web. The first and perhaps easiest way is to use the Servvoy Web Client. There is no code required – you can point end users to a launch page showing all the available solutions (somewhat similar to an “Open” dialog) by pointing to: <http://yourIP/servvoy-client>. Users can then click on a particular solution and Servvoy will automatically render the application in pure HTML and CSS; as well as leverage AJAX (Asynchronous JavaScript and XML) to dynamically update the web page without page refreshes – all without you/the Servvoy developer having to write any code. In addition, if you would like to customize your web pages outside of Servvoy, you can edit the pages with an external HTML editor, like Dreamweaver, and add your own custom HTML, CSS, Javascript, or external components.

You can also use the Servvoy Headless Client API and JSP (Java Server Pages) to access your solution. This is really a great way to write custom web applications while sharing the same business logic as the Servvoy Smart Client application. Just include an eight line wrapper, and you can access your solution’s methods and data; and you can pass your methods parameters from the browser and receive back HTML, data – even Java objects like recordsets, binary data and more.

Because the Servvoy Headless Client API knows how to connect to your solution, you don’t have to specify a database connection, or a table name, or a SQL query. By developing your methods in Servvoy, you can leverage Servvoy’s step debugger when developing your methods – and re-use existing business logic and validations.

With Servvoy, all printing automatically goes through PDF (without any extra coding) – making it very straightforward to print perfectly to the pixel with proper page breaks. The PDF is generated on the server, and then sent down to the client providing consistency across all users, platforms and browsers.

## **Maintaining and changing your application**

### *Making Changes to a .NET Application*

If the developer needs to make changes to an existing application, the .NET framework doesn’t provide any automated way of deploying the changes to all users. Using Visual Studio, you must

compile and re-export your solution as a Windows Installer file. From there, it is up to you to handle updating the application on each client. You must either build your own deployment solution, or purchase a third-party solution (if available).

For web applications in ASP.NET, you have to make changes to the .asp files and re-upload them to the web server.

One of the biggest problems in this area is the dependency checking. It is unclear when an update to code is made, whether the code change may or may not affect other parts of the application. Because of this issue, extensive testing procedures have to be in place to ensure that new updates don't break existing functionality.

For both desktop rich client and web applications, the .NET framework doesn't provide versioning tools, so you can't automatically update or rollback to a version if there is a problem. This must be handled manually.

### *Making Changes to a Servoy Application*

With Servoy, you simply export your solution out of the development repository – using Servoy Developer – then import it into the production repository via a web browser (regardless of where the server is physically located). You can then activate the new release with or without clients connected, and you can force the clients to re-load the solution to get the new version, or simply continue using the older version. It's up to you. This process will update the rich client and web client application at the same time – automatically. There is no need to restart the application server to roll-out new versions of your application.

Servoy also allows you to roll-back to a previous release in production without having to re-upload an older version. The application server has a built-in versioning system to maintain older versions of the applications on standby for redeployment when necessary.

## Upgrading the Applications

### *Upgrading .NET*

From time to time, Microsoft will release an update to their .NET framework. The framework must be upgraded on each user's machine. The .NET framework doesn't provide any automated way of doing this, so the developer must do each install manually, or purchase a third-party tool (if available). In many cases, being forced to update the .NET framework for each user will also require rewriting large parts of the application in order to use new functions made available. This shortcoming is probably due to the fact that .NET is a relatively young and immature framework. Java being 7 years older, does not suffer from these problems. For example: For over ten years, JDBC has been the standard way to connect to a database in Java; but with .NET, the way to connect to a database has been changed more than 4 times since the .NET framework was originally introduced.

### *Upgrading Servoy*

Upgrading Servoy is very easy. Servoy Developer has a self-updating mechanism built-in. If there is a new version of Servoy, you are automatically prompted to download and update to the newer version when launching the application.

Updating Servoy Server is easy – simply run the server updater and restart the application server (or service or daemon). That's it. Servoy is completely self-configuring and previous versions of your solutions will continue to work.

## Licensing Methods and Costs

### *.NET Costs*

The installation of the .NET runtime environment is free. Microsoft SQL Server 2005 Enterprise edition is \$24,999 per processor. The Visual Studio developer environment is \$2,499 per developer. Since .NET applications only run on Windows, you must purchase a Windows Server license, Terminal Services (TS) and a CAL (Client Access License) for each client connection. The following example assumes a 20-user connection to a rich client application over a WAN, which requires TS licenses.

## Servoy Costs

Servoy is licensed per concurrent user, not per seat – meaning if you have 100 people in your organization, but only 85 will use the application at one time, you only need to purchase 85 licenses. You can choose to deploy to Servoy Smart Client or Servoy Web Client (or mix and match) from the same Servoy Server at the same time. The cost per each concurrent client is \$289, and there is NO additional charge for Servoy Server – and NO additional charge for the Sybase iAnywhere SQL database engine (when used with your Servoy application). Servoy Smart Client is low enough on bandwidth for WAN users that you don't need to purchase any additional terminal service or Citrix licenses.

Both Servoy and Microsoft .NET offer discounts for purchases of multiple licenses at the same time. But, when you take into consideration licensing costs, Servoy is almost 66% less expensive than .NET.

NOTE: The example shown below only takes licensing costs into account. Including all costs of development, deployment and maintenance – TCO, this difference will be even more significant.

<i>Item</i>	<i>Servoy</i>	<i>.NET</i>
Client Licenses (100 for .NET, 85 for Servoy)	\$19,465	\$0
Database Server License	\$0 (Sybase included)	\$49,998 (SQL Server)
Developer Licenses (5)	\$3,245	\$12,495
<b>Software License Total</b>	<b>\$22,710†</b>	<b>\$62,493†</b>
New Production Server**	\$3,188	\$3,188
Microsoft Windows Server Enterprise Edition*	\$0	\$3,999
Microsoft TS for Wan Users (20)	\$0	\$2,979
Microsoft Windows CALs (80)	\$0	\$3,196
<b>Hardware/OS License Total</b>	<b>\$3,188</b>	<b>\$13,362</b>
<b>GRAND TOTAL</b>	<b>\$25,898</b>	<b>\$75,855</b>

†All prices in US dollars. Prices based on published prices from both web sites as of March, 2007.

\*.NET requires a Windows server

\*\* Prices are based on a Dell PowerEdge 2900 with (2) Dual Core Intel® Xeon® 5130, 4MB Cache, 2.00GHz, 1333MHz FSB, 2GB 533MHz (4x512MB), Single Ranked DIMMs, 36GB, SAS, 3.5-inch, 15K RPM Hard Drive, Microsoft Small Business Server 2003 Standard Edition – pricing as of March 2007.

## Conclusion

From an organizational perspective, the .NET framework is most suited for companies who are completely Windows-based; and don't need to worry about cross-platform compatibility. From a development perspective, the .NET framework is most suited for developers who prefer to write their own frameworks completely from the ground up. If you are a company or developer who requires cross-platform applications, ease of installation and deployment, and a development framework that is already written to allow you to rapidly create your applications, then the .NET framework falls short of your needs. Servoy is the answer to your problems.

Servoy allows you to build cross-platform applications with a single code base, and effortlessly deploy your applications over a WAN and LAN via the rich desktop Servoy Smart Client, or through a web browser with Servoy Web Client. Servoy comes with a framework to help you quickly create your applications and connect to the database(s) of your choice. And you can still extend your application by integrating with other technologies (including Microsoft's) using the Servoy Java API. With Servoy, you get all of these features in a standards-based application that costs less, and allows you to build your enterprise-class applications faster than with the .NET framework.

The approach of both environments is very different: .NET tries to be a Windows-only version of the traditional Java platform with some small enhancements in terms of nicer user interfaces and slightly better productivity than Java programmers. Servoy's approach is radically different: Servoy has been designed from the ground up to make programmers more productive and to allow companies to deploy and maintain their applications at a lower cost

## About Servoy BV

Servoy BV is a privately-held company established to develop, sell and support the Servoy suite of products. The idea for Servoy started in 1998 by the four co-founders of the company -- being frustrated with the limitations of desktop database tools on one hand; and the complexity of web-based development tools, the steep learning curve, and the long development time on the other.

Today over 1500 companies and more than 10,000 developers are working with the Servoy suite of products. Companies like Symantec; Stanford University; Verizon; and UCLA hospital rely on Servoy for managing and presenting data to their customers and employees – providing rich applications over LAN, WAN and Internet connections. Servoy can count Apple, Oracle and Sybase among its technology partners.

The Servoy worldwide headquarters is located in The Netherlands (Amersfoort) where all research and development; as well as international sales and marketing activities are centralized and coordinated; Servoy Inc., Servoy's US office, is responsible for all US and North American sales and marketing events.

### International Headquarters

Servoy B.V.  
Algolweg 9A  
3821 BG Amersfoort  
The Netherlands  
Voice: +31 33 455 9877  
Fax: +31 84 883 2297

### Servoy USA

Servoy Inc.  
299 W. Hillcrest Drive  
Suite 115  
Thousand Oaks, CA 91360  
Voice: (805) 624-4959  
Fax: (805) 624-4958

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, AND SERVOY BV DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, OR THE WARRANTY OF NON-INFRINGEMENT. IN NO EVENT SHALL SERVOY OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER INCLUDING DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, LOSS OF BUSINESS PROFITS, PUNITIVE OR SPECIAL DAMAGES, EVEN IF SERVOY OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY. SERVOY BV MAY MAKE CHANGES TO THIS DOCUMENT AT ANY TIME WITHOUT NOTICE. THIS DOCUMENT MAY BE OUT OF DATE AND SERVOY MAKES NO COMMITMENT TO UPDATE THIS INFORMATION.

©2007 Servoy, Inc. All rights reserved. Servoy is a trademark of Servoy, Inc., registered in the U.S. and other countries. All other trademarks are the property of their respective owners. Product specifications and availability subject to change without notice.