# ActiveADAPTER

# Top Ten BizTalk Active Directory Integration Mistakes

## 1. Ignoring the importance of the retry, redundancy and other native features of ports and adapters.

Much has been written on this topic. Check out Erik Westermann's comments, for example, at http://artofbabel.com/columns/top-x/49-top-10-biztalk-server-mistakes.html

Ports and adapters are the basis of BizTalk's agility and power. Use them whenever possible.

## 2. Underestimating the complexity of Active Directory programming and data types.

Many Active Directory object properties are single-valued strings and integers. But a lot are multi-valued. Others are byte arrays, datetime stamps, bitwise flags... And then there are GUIDs, SIDs…

There are other complexities too. For example, to set a password or add an object to a group you don't write to a property. Instead you call a method of the object.

The best solution is to use an adapter. If you decide not to use an adapter, however, recognize that you'll need significant time and skills to architect a solution and get BizTalk connected to Active Directory.

Don't put the budget and timing of your project in jeopardy. Plan and resource well.

## 3. Not seeing all the high ROI dimensions of BizTalk-Active Directory integration

User provisioning is the slam dunk return on investment opportunity that gets us developing BizTalk to Active Directory applications. But there are plenty of other great opportunities too. Like Active Directory:

- Auditing (especially security auditing)
- Grooming
- Instrumentation (need an alert sent when someone is granted access to a resource?)
- Synchronization

An ActiveADAPTER customer recently used BizTalk to create a system that allowed users to apply for administrative rights on their local machines for an hour. The entire process including the approval process, granting and rescinding the admin rights, and logging the request was handled by BizTalk.


©2011 ActiveADAPTER

You've got BizTalk. You've got Active Directory. Connect the two and there's very little you can't do to manage your organization.

## 4. Failing to realize that not just the content but the STRUCTURE of AD can change

Companies merge. OUs get moved and renamed. Security requirements change. Your BizTalk Active Directory application needs to be agile. Most importantly, you need an easy, reliable way to alter LDAP binding strings so you can refactor your BizTalk application to connect to the right points in AD when the structure changes.

## 5. Not including indexed properties in your query

Active Directory indexes some object properties like `objectCategory` by default. If you include these in your queries you will get faster performance. (In fact, this is why the ActiveADAPTER Send Adapter emphasises the "ObjectFilter" parameter.)

For example:

```
(&(objectCategory=user)(cn=Jane Active))
```

Will be faster than just:

```
(cn=Jane Active)
```

Different properties are indexed in different versions of Active Directory so check your documentation. You can also configure your own indexed properties.

## 6. Making security unmanageable by diffusing it through your code

So often BizTalk to Active Directory development goes like this: First you hard-code a set of credentials to connect to your development AD domain. Then later you realize you need to use another set of credentials to connect to a second domain. Then it's time to move to test, so you rework your orchestration to put the credentials (in plain text!) in the BTS Config file. You change the values in the config file again for the move to production. Unfortunately, six months later the password gets changed for the account you used. It takes a week to figure out why your BizTalk application has suddenly stopped working.

Odds are you will also need to hard code domain controller names for server binding in your LDAP strings. But that's another story…

Think about credentials and migration from development to test to production (and beyond) as part of your initial planning.


©2011 ActiveADAPTER

## 7. Putting too much code in Expression shapes

We all know how easy it is to fill up Expression shapes with a LOT more code than they were intended to hold. (In fact, many developers think that the Expression Shape code box was not resizable prior to BizTalk 2010 to prevent over-coding). Even if you've slogged through creating and deploying your own custom .NET assembly, manipulating it in code in your orchestration soon gets very ugly and hard to maintain.

Keep code within orchestrations to a minimum.

## 8. Getting too "point to point" focussed

This generally leads on from too much coding. You end up focussing on achieving a system that works for ONE scenario instead of ALL scenarios.

Look at the BizTalk ESB patterns and EAI patterns in general for optimal ways to move messages through BizTalk. As soon as you focus too precisely on just creating a system that works you can lose sight of loose coupling and all the powerful features BizTalk provides.

## 9. Scalability

All the mistakes listed above can paint you into a corner as far as scalability is concerned.

If your queries start returning too many results, can you easily add a new port and distribute the load?

If you need to update a new domain (or even a new forest), how will your system cope?

What if your organization moves to using Active Directory Branch Office services?

These nightmares can be avoided by using the right tools and the right application design.

## 10. Not planning for system evolution into the cloud

Cloud computing is a paradigm shift. Technologies such as WCF, AD Federation Services and Azure Integration Services will affect almost every organization.

Access Control Services (ACS) are a big piece of the cloud jigsaw and Active Directory is likely to be a big part of your ACS strategy. Consider how you might like your BizTalk to Active Directory applications to position you to exploit the cloud.


©2011 ActiveADAPTER