## **RIA SERVICES INVOKE OPERATIONS**

John Sadd Fellow and OpenEdge Evangelist Document Version 1.0 February 2011





## DISCLAIMER

Certain portions of this document contain information about Progress Software Corporation's plans for future product development and overall business strategies. Such information is proprietary and confidential to Progress Software Corporation and may be used by you solely in accordance with the terms and conditions specified in the PSDN Online (http://www.psdn.com) Terms of Use (http://psdn.progress.com/terms/index.ssp). Progress Software Corporation reserves the right, in its sole discretion, to modify or abandon without notice any of the plans described herein pertaining to future development and/or business development strategies. Any reference to third party software and/or features is intended for illustration purposes only. Progress Software Corporation does not endorse or sponsor such third parties or software.

The videos and papers in this series on using Silverlight with RIA Services have shown examples of passing a ProDataSet as a parameter and converting it to an entity class that can provide data to a Silverlight user interface. This paper and the two-part video it accompanies show you an example of how to call an ABL procedure that doesn't pass a ProDataSet or temp-table, but instead passes one or more scalar or array values as input and output parameters. Earlier examples in the series show the **Query** and **Update** attributes as annotations in the DomainService class. The **Query** example just has a ProDataSet as an ABL **OUTPUT** parameter. The AppObject proxy expresses this as an ADO.NET DataSet, and the DomainService query method itself defines it as an enumerable collection of **CustOrder** objects, as shown here:



An example such as that one could be extended to have one or more **INPUT** parameters as well, specifying filter criteria for example. But the **Query** form can't have any output parameters other than the DataSet, or alternatively a temp-table.

The **Update** example in the RIA Services series passes in just a changed row to return to the server, as shown here:

[Update]
<pre>public void UpdateCustOrder(CustOrder changeRow)</pre>
{
DataRow updateRow = FindCustomer(changeRow.CustNum);
if (updateRow != null)
{
updateRow["CustNum"] = changeRow.CustNum;
}
}

The associated submit method passes this as an ADO.NET change set back to the server. The Update and Submit form in fact doesn't allow additional parameters beyond the change set. As a reminder, the Submit method code below shows both the Submit call with the object-oriented change set parameter and then the AppObject proxy call that passes the DataSet form of that change set through the proxy to the ABL UpdateCustOrders procedure:



There is also a general purpose attribute named Invoke for making a call that doesn't involve an entity, but needs to pass some combination of input and output parameters, which can be scalar values or arrays of scalar values. That's what is covered in this paper and the two-part video that concludes the RIA Services series.

As in the earlier code samples in the series, this one defines an ABL service as a stand-alone non-persistent procedure, which takes a customer number as input, retrieves the corresponding SalesRep record from the database, and returns the SalesRep name as output:

```
/*-----
File : GetSalesRep.p
Notes : Returns the SalesRep RepName for the given customer number.
DEFINE INPUT PARAMETER piCustNum AS INTEGER.
DEFINE OUTPUT PARAMETER pcSalesRepName AS CHARACTER.
FIND Customer WHERE Customer.CustNum = piCustNum.
FIND Salesrep WHERE Salesrep.SalesRep = Customer.SalesRep.
pcSalesRepName = Salesrep.RepName.
```

This is a somewhat simplified case. If your ABL procedure has a single **OUTPUT** parameter, and doesn't need to return the ABL **RETURN-VALUE** to the client, then ProxyGen maps the one **OUTPUT** parameter to the return value of the **Invoke** operation in the DomainService class, which this paper shows how to define. The

procedure could still have one or more **INPUT** parameters, and the process would be the same. However, if the procedure has multiple **INPUT-OUTPUT** or **OUTPUT** parameters, then there's an additional code construction that's needed, and an example of that is shown in the second of the two videos, and described in the second part of this paper.

So again, the newly-created the ABL procedure needs to be added to the same proxy as the other procedures that represent services on the Customer and Order tables. Opening the proxy that was built and extended before, I can add this next procedure to it. Like the others, it's in the AppServerContent folder, because that's what gets published to the AppServer's ProPath. Selecting the .r file for the new procedure, I add that to the proxy:

Proxy Generator for	or Open Clients - C:\WebWorkspace\SilverlightSample\CustOrderApp.xpxg	
File SubAppObject Pr	rocedure Tools Options Help	
🔊 🏂 📓 🙎	<b>b</b>	
CustOrderApp	AppObject Procedures	
	Non-persistent Procedures (2)	
	C:WebWorkspace\S Add Non-persistent Procedures for EustOrderApp	
	Propath Selection C:\WebWorkspace\SilverlightSample\ Use Path in Name	
	File Filter "r Perform Recursive Add	
	Moye to F C:\WebWorkspace\SilverlightSam MGetSalesRep.r Add	
	settings	
	Persstent Procedures	
	EustOrdersNS Help	
	SaveRead	
J		
	I I I I I I I I I I I I I I I I I I I	
	0 directories selected; 1 files selected 2 Procedures	
	·	
		11.

Once again I re-generate the proxy, and now I have a new version with three separate procedures defined.

Next I have to add a reference to it in the DomainService class. Back in Visual Studio, I re-open the **CustOrderDomainService** class. Down at the bottom of the file, I add support code for this first non-entity-passing method, which will run **GetSalesRep** in the proxy, and which as I said uses an attribute named **Invoke** to signal the behavior that needs to be supported. The **Invoke** attribute is expressed as an annotation in brackets, and the new public method has the same name as the ABL procedure that it's going to run. (This is just a naming convention I use for this example; there's no need to use the same name, and in fact you might prefer not to.)



The method just takes the input parameter that will be passed to the ABL procedure, the customer number. I need to define a variable to hold the output parameter that comes back from the ABL procedure. Then in the same AppObject instance used by

other methods in the DomainService, the GetSalesRep method needs to invoke the ABL procedure through the proxy. This call takes the parameters of the ABL procedure: the customer number as input and the SalesRep name as output. So the DomainService method returns the SalesRep name as its return value:

	<pre>string outRepname;</pre>
	<pre>appObj.GetSalesRep(inputCustNum, out outRepname);</pre>
	return outRepname;
}	

That's the end of the new method in the DomainService. After saving and compiling the DomainService, I do a Build to regenerate the DomainContext class in the solution. Once that's done, I need to update the user interface to display the SalesRepName that's returned from OpenEdge. Back in the XAML for the **CustOrderPage**, I need to add a button that will trigger a request to retrieve the SalesRep name. In the Toolbox, I select a Button control:

00	CustOrde	erProject - Microsoft Visual Studio							
Fib	e Edit	View Project Build Debug Team Data 1	ools Test V	Window Help					
1	in - 🔛	- 🗠 - N - C - 🖓 🖬 🕺 🕷 🖬 🛀	- 🖳 🕨 🛛	Debug -	getsalesrep	- 🖓 🕾	* 🕠	🕺 🎠 🛃 🛃 🗉 = 🖕	
1	B & 9	5 사 個 建美 물 일 🗆 위 위 위	1 G4 68 1	😡 🚽 Publish	Create Publish Settings 🚽	s, e =			
*	Toolbox	* ⊕ X					-	Solution Explorer	- 4 ×
ā	Cust(	OrderProject Controls						🕒 🔉 🛃 🖧 🌮	
Ř	<u></u>	Pointer						🕂 🗁 Models	
	巴	BusyIndicator (CustOrderProject.Controls)	^					🗄 📄 Shared	
묥	<b>B</b>	CustomDataForm (CustOrderProject.Controls)						Custorder.cs     RegistrationData.cs	
Š	巴	ErrorWindow (CustOrderProject)						User.cs	
S.	巴	LoginForm (CustOrderProject.LoginUI)				0		🗄 - 🗀 obj	
ι.	巴	LoginRegistrationWindow (CustOrderProject						Resources	
	۳.	LoginStatus (CustOrderProject.LoginUI)						Gervices     Gervices     Gervices	s 📕
	巴	MainPage (CustOrderProject)						🥶 CustOrderDomainServic	e.cs
	巴	RegistrationForm (CustOrderProject.LoginUI)	0		Animation			🕙 UserRegistrationService	e.cs 💌
	Comn	non Silverlight Controls			(1024 x 680) Submit Changes			Sak ting European Tana European	
	R.	Pointer			(X:0; Y:0)			- Soldon Explorer	
		Border						Properties *	- 4 ×
	æ	Button 🛇						DataGrid CustomerGrid	
		CheckBox						Properties / Events	
	÷.	ComboBox							
	<b>j</b>	DataGrid						Search	×
	<u> </u>	Grid			=			CanUserSortCol 📮 🔽	
	1	Image	gnHeight=	480"			÷.	CellStyle Resource	
	Α	Label	Xmins:sd	K="nttp://so	nemas.microsoft.com/winf	rx/2006/xam1/pro		Cip 12	1
	÷0	ListBox	umns="True	e" Height="1	37" HorizontalAlignment=	="Left" Margin="	-	CinboardCopyM III Excludettear	
	۲	RadioButton	lumns="True	e" Height="1	.00" HorizontalAlignment=	="Left" Margin='	-	ColumnHeaderH II NAN	
		Rectangle	es" Height	t="23" Horiz	contalAlignment="Left" Ma	argin="464,161,0		Columnities and arS III Decourse	- 1
		StackPanel						Column Color D	1
		TabControl						(Collection)	<u> </u>
	📸 Erro								h

I drag and drop that onto the page, and move it over to the left, underneath the Customer grid. The screenshot below shows that the XAML now includes the Button and reflects its initial property settings:

o CostOrdenBusicat Minners B. Hinnel Charlie		
Elle Edit View Drolect Build Debug Team Date Engrat Tools Test Window Hein		
The car way indicated and being found bands former food for white indicated and the set of the set		
· · · · · · · · · · · · · · · · · · ·		
CustOrderDomainService.cs CustOrderPage.xaml* ×	Solution Explorer	≁ ù ×
§ 10031	🕒 📴 🖬 🖧 🎾	
Armation Submit Changes (1024 x 669)	CustOrde     Guidente     Guidente	r.cs nData.cs ationService.cs trationService.cs trationService.cs Tesm Explorer ↓ ×
	Properties Properties Even	nts X
L Design 14 E XAML	Content 🔶	Button
<pre><sdk:datagrid <="" autogeneratecolumns="True" height="100" horizontalalignment="Left" margin="" pre="" true"=""></sdk:datagrid></pre>	ContentTemplate	Resource
<Button Content="Submit Changes" Height="23" HorizontalAlignment="Left" Margin="464,161,0</th> <th>Cursor 🛛</th> <th></th>	Cursor 🛛	
	DataContext 🛛	Binding
<pre>[</pre>	Effect 🛛	Value must be se
100 % - 4	FlowDirection	LeftToRight
Button (button1) Page/Grid/Button	FontFamily 🛛	Portable User 💌
Ready		h

I change the button name to **SalesRepButton** in the Properties tab, and change the label – that's the **Content** property -- to **Get SalesRep**. I can resize the button as well to make room for the new label:

ः CustOrderProject - Microsoft Visual Studio	
File Edit View Project Build Debug Team Data Format Tools Test Window Help	
i 📴 • 🔛 • 😂 📕 🌒 🙏 🖧 🖄 🖑 🕫 - (* - 💭 • 🖏 🕨 Debug 🔹 📴 getsalesrep 🔹 🔹 🖓	🐋 🏷 🛃 🖳 e 🖕
同名と * 作  崇拝  三名  ロタマ 和母名 みの, Publish: Create Publish Settings 🚽 🕺 🖉 🗸	
💋 CustOrderDomainService.cs CustOrderPage.xaml* 🗙	Solution Explorer 👻 부 🗙
8 1000	🗟 🔯 🖬 🖧 🎓
	AuthenticationService.cs     GustOrderDomainService.cs
Get SalesRep	UserRegistrationService.cs
	Solution Explorer
	Properties 👻 🕂 🗙
	Button SalesRepButton
	Properties 🥖 Events
	Search ×
Design 14 © XAML LINE NUCOGENERALECOLUMINSE TIME RELETION ROLLZONCALALIZAMENCE LETIC MARGINE	Content 🔶 Get SalesRep 💻
<sdk:datagrid <br="" autogeneratecolumns="True" changes"="" height="23" horizontalalignment="Left" margin="464,161,0,0" submit=""><button <="" content="Get SalesRep" height="23" horizontalalignment="Left" margin="42,161,0,0" th=""><td>Cursor 🖬 🚽</td></button></sdk:datagrid>	Cursor 🖬 🚽
	DataContext 🛛 Binding 🖌
<pre> </pre>	Effect 🛛 Value must be se
100 % - 4	FlowDirection 🛛 LeftToRight
Button (SalesRepButton) Page/Grid/Button V	FontFamily 🖸 Portable User 💌
ing Error List Roady	

Now I want what ABL would call a fill-in field to display the SalesRep name that's retrieved. Once again in the Toolbox, I select a **TextBox** control, and drag that onto the page, and move it over next to the button. I name it **RepnameBox**.

○ CustOrderProject - Microsoft Visual Studio	,	. [D] ×
File Edit View Project Build Debug Team Data Format Tools Test Window Help		
🖏 • 🛍 • 🥁 🖟 🥔 🕺 🖄 🖏 🖏 🔹 • • • 🖓 • 🖏 🕨 Debug 🔹 🧭 getsalesrep 🔹 🕫 🖓 🕾 🕻	💌 🛠 💽 💁 🖘 🖕	
김 원원 🗤 作 律律 글 일 🗆 위 두 위 두 위 등 및 🚦 Publish: Greate Publish Settings 🕞 및 🖕		
CustOrderDomainService.cs CustOrderPage.xami* X	Solution Explorer 🗸 🗸	φ×
100%	🔁 🔯 🖬 🖧 🐌	
	🖻 - 📴 Models	
	Guttorder cr	
	RegistrationData.cs	
	🕙 User.cs	
	Contraction (1970)     Contraction (1970	
	Services	
	AuthenticationService.co	s
	💷 🥶 UserRegistrationService	.cs 💌
Get SalesRep Submit Changes		
	- Soldton Explorer	
	Properties	4 ×
	Button SalesRepButton	a la sette
	Properties 🐓 Events	_
	Search Search	×
	Content 📀 Get SalesRep	P 🔺
<sdk:datagrid <="" autogeneratecolumns="True" changes"="" get="" height="23" horizontalalignment="Left" margin="42,161,0,0" sale:ren"="" submit="" th=""><th>Cursor 🛛</th><th>_</th></sdk:datagrid>	Cursor 🛛	_
<textbox height="23" horizontalalignment="Right" margin="0,161,379,0" name="RepnameBox" th="" v<=""><th>DataContext 🔲 Binding</th><th>4</th></textbox>	DataContext 🔲 Binding	4
	Effect 🔲 Value must b	e se
100 % * 4	FlowDirection 🔲 LeftTaRight	_
Button (SalesRepButton) Page/Grid/Button	FontFamily 🔲 Portable Use	r 💌
🙀 Error List		
leady		4

The value displayed is the control's **Text** property. The supporting code for the UI sets that at runtime when a SalesRep name is retrieved for a Customer.

If I double-click on the button, I get the skeleton for a **Click** event handler for it. Here I want to retrieve the customer number from the currently selected row in the grid to use as the input parameter to my call:

00 EustOrderProject - Microsoft Visual Studio		_O ×
File Edit View Refactor Project Build Debug Team Data Tools Test Window Help		_
👔 😳 * 📴 * 🎯 🔜 🕼 🐇 🔩 🥙 * 🖓 * 🖓 * 🖏 🕨 Debug 🔹 🧭 getsalesrep 🔹 💫	2 🕁 📑 📷 🎋 🛐 🐻 🗤	* -
🖪 勉 🌭 ** 愼 課 課 📃 월 💷 🖓 🤤 🖓 🖓 🎝 🎝 🎝 💭 📮 Publish: Create Publish Settings 🔷 🖧 🧭 🖕		
CustOrderPage.xaml.cs* × CustOrderDomainService.cs CustOrderPage.xaml*	<ul> <li>Solution Explorer</li> </ul>	+ 4 ×
💈 🔩 CustOrderProject. Views. CustOrderPage 🔹 😵 SalesRepButton_Click(object sender, RoutedEventArgs e)	🗸 🗟 🙆 🗸	1
<pre>CODomainContext.SubmitChanges(OnSubmitCompleted, null); }</pre>	⊕ → Models     ⊕ → → Models     ⊕ → → → Sha     ⊕ → ↓ → ↓ ↓ ↓     ↓	red tOrder.cs
<pre>private void OnSubmitCompleted(SubmitOperation submitOp) {     if (submitOp.HasError)</pre>	⊞ (≦) Use	istrationData.cs r.cs
<pre>{</pre>	e- in Resource B- in Services Aud Cus Use	res henticationService.cs tOrderDomainService.cs rRegistrationService.cs
<pre>private void SalesRepButton_Click(abject sender, RoutedEventArgs e) {</pre>	Solution Explorer	Team Explorer ▼ 및 ×
	31 <mark>24</mark> 🖻	
	-	
100 % - 1	×	
📸 Error List		
Ready Ln 131	Col 13 Ch 13	INS //

I start with the same statement used in the **SelectionChanged** event that retrieves Orders for a selected customer. **SelectedCust** is a **CustOrder** entity from the **CustomerGrid**, and the current row is the **SelectedItem** property:

private void SalesRepButton Click(object sender, RoutedEventArgs e) { CustOrder selectedCust = (CustOrder)CustomerGrid.SelectedItem;

Now what comes next is a little tricky. Remember that the user interface C# support code doesn't make calls directly to the DomainService. It uses the generated DomainContext class as a client-side intermediary. If I open that class once again in the **Generated\_Code** folder (which I exposed in an earlier video by pressing the

**Show All Files** button) you can take a look at the code generated to support GetSalesRep. Look at the comments for the GetSalesRep operation:

/// <summary></summary>
/// Asynchronously invokes the 'GetSalesRep' method of the DomainService.
///
/// <param name="inputCustNum"/> The value for the 'inputCustNum' parameter
of this action.
/// <param name="callback"/> Callback to invoke when the operation
completes.
/// <param name="userState"/> Value to pass to the callback. It can be
<c>null</c> .
/// <returns>An operation instance that can be used to manage the</returns>
asynchronous request.
<pre>public InvokeOperation<string> GetSalesRep(int inputCustNum,</string></pre>
Action <invokeoperation<string>&gt; callback, object userState)</invokeoperation<string>
{
Dictionary <string, object=""> parameters = new Dictionary<string,< th=""></string,<></string,>
<pre>object&gt;();</pre>
parameters.Add("inputCustNum", inputCustNum);
this.ValidateMethod("GetSalesRep", parameters);
return ((InvokeOperation <string>)(this.InvokeOperation("GetSalesRep",</string>
typeof(string), parameters, true, callback, userState)));
}

First there's a reminder that calls to the DomainService are made asynchronously, so the code has to be a two-step process of making the request and then processing the response. The first parameter is my input parameter, the customer number. Then I have to name a callback method that's run when the request completes. Then there's an optional user state value that I'll just set to null; this is a pattern we've seen before. Most importantly, you can see that the call itself returns an InvokeOperation object.

Looking at this generated code in the DomainContext confirms what the call in the user interface support class needs to look like. I have to define an InvokeOperation object of type string, representing the one output parameter, which I call invokeOp. I assign that to the result of the method in the DomainContext called GetSalesRep. The first parameter to the method in the DomainContext is the customer number. To get to that, I start with the SelectedCust in the CustomerGrid, which is an instance of the Custorder entity, and so has values for all the fields that are defined as properties in the entity class. I want the CustNum property, of course. You can see the code assist is reminding me that the next parameter is a callback method:

🗢 EustOrderProject - Microsoft Visual Studio	<u>_</u> _×
File Edit View Refactor Project Build Debug Team Data Tools Test Window Help	
i 🛐 - 🛅 - 🎽 🚽 🗿 🕺 🔄 🖏 🦘 - 🔍 - 💭 - 🖏 🕨 Debug 🔹 🦉 GetSalesRep 🔹 🗞	🕾 🖓 🖄 🎋 🛃 📓 🗉 📲
1 🕄 🖄 🌭 🗚 🎁 🐺 🗒 🚊 ڬ 💭 🗣 😂 🖓 🖓 😓 🦓 🖕 Publish: Create Publish Settings 🔷 🔌 🖉 🖕	
🔏 CustOrderPage.xaml.cs* 🗙 CustOrderDomanService.cs CustOrderPage.xaml	✓ Solution Explorer
💈 🔧 Cust Order Project. Views. Cust Order Page 🔹 🚽 Sales Rep Button_Click (object sender, Routed Event Args e)	🗸 🕒 💽 🖉 🖉 🗸
<pre>CC00mmainContext.SubsitChanges(OnSubmitCompleted, mull); } CC00mmainContext.SubsitChanges(OnSubmitCompleted, mull); } private void OnSubmitCompleted(SubmitOperation submitOp) {     ff (submitOp.HasError)</pre>	
Invo callback: Calback to invoke when the operation completes.	
CODomainContext.GetSalesRep(selectedCust.CustNum, ) ) 100 % • 4	21 91 ···
😼 Error List	
Ready Ln 134	Col 17 Ch 17 INS

I call that callback OnSRB\_Completed - SRB for **SalesRepButton**. I can just use null in place of a user state, and that's the end of the click event handler:



Now I need to code the callback method, OnSRB\_Completed, which takes the InvokeOperation object as input:



The method uses the same HasError check used in other callbacks like this one. Otherwise, if there's no error, all the code needs to do is assign the Text property of the RepnameBox TextBox to the Value property of the Invoke operation, which is a string. The Value property is the return value that the ABL procedure's output parameter is mapped to.

Just to clean up the display, I make a small change to the SelectionChanged event handler that gets fired every time the user selects a different customer in the grid. I clear the Text value of the TextBox, until the user clicks the **Get RepName** button again:



Of course, the code could trigger the GetRepName method from this SelectionChanged event handler, so that the value is immediately displayed whenever a new Customer row is selected, but that's just part of the user interface design, so feel free to make a change like that in your own examples.

Now I can run the test page for the project. I select the CustOrder page, select a Customer, and click the Get SalesRep button. Here's the SalesRep name for that Customer:

CustOrderF	Page Page - Wi	ndows Internet Explo	rer							=D×
•	http://localh	ost:52878/CustOrderPr	ojectTestPage.asp	#/OustOrde	rPage			🗉 🔂 🐓 🗙 🚼 Google		. م
Favorites	🍰 💋 Sugge	sted Sites 🔹 🙋 Web :	Sice Galery 🝷							
CustOrderP	Page Page							🔓 • 🗟 • 🖬 👼 • I	Page + Safety +	Tools * 😧 * *
🔁 Ap	plication N	lame						Home	About	
			_	_	_		-			login
Custilium	No. 10	0 didaaa	city.	Finhs	Condition	- it				
Custivum	Name	Address	City	State	CreditLi	nit				
1	Lift Tours	276 North Dr	orie Dunington	MA	00/00					
2	Hoops	Euite 415	Atlanta	GA	25000					
з 4	Go Eiching Li	d Unit 7	Marrow	Middlerey	15000					
5	Natch Roint 3	Cennis 66 Homer Pl	Boston	MA	11000					
Cat	CalarRan Dit	b Diek K				ubmit Channer				
Oct :	baleskep Pic	C / DIIK KI				submit changes				
CustNum	OrderNum	Carrier	OrderDate		SalesRe	p				
2	94	Standard Mail	10/4/1997 12:	MA 00:00	DKP		-			
2	125	FlyByNight Courier	12/20/1997 12	2:00:00 AM	DKP		1			
2	171	Standard Mail	11/30/1997 12	MA 00:00:2	DKP					

I can select another Customer, and you can see that the SalesRep name was cleared by the line of code that I added to the **SelectionChanged** event handler:

usturden	rage Page - W	noows internet	capaorer							
) <del>-</del> ()	http://local	host:52878/CustOr	derProjec	tTestPage.asp	≪#/CustOrd	rPage		_	🛨 🔂 🐓 🗶 🔀 Google	٩
Favorites	👍 🄏 Sugge	ested Sites 💌 🙋	Web Slice	Gallery 🕶						
CustOrderF	age Page								🏠 • 🖸 - 📑 📾 • Page • Safety	• Tools • 😥 •
•										
🌖 Ар	plication I	Name							Home About	CustOrder
										login
CustNum	Name	Address		City	State	Credit	imit			
	Lift Tours	276 Nor	th Drive	Burlington	MA	66700				
2	Urpon Frisbe	eet Rattipol	ku 33	Oslo	Uusima	27600				
3	Hoops	Suite 41	5	Atlanta	GA	75000				
	Go Fishing L	td Unit 2		Harrow	Middlesex	15000				
5	Match Point	Tennis 66 Hom	er Pl	Boston	MA	11000		•		
Get 5	SalesRep						Submit Changes			
CustNum	OrderNum	Carrier	OrderD	ate	Sales	Rep				
	21	Standard Mail	2/8/19	98 12:00:00	AM SLS					
	6030		2/3/19	98 12:00:00	AM BBB					

So I click the button again, and I see the SalesRep for customer 4, so everything's working:

Intro / Machada 5287% (CutOrderProgetTedProge. sapuré) (CutOrderProgetTedProge. sapuré) (CutOrderProgetTedProge. sapuré) (CutOrderProgetTedProg	p
Føreter	Nome About CustOrder
Custinum Name     Address     City     State     CreditLimit       1     Lift Tours     276 North Drive     Burlington     NA     66200       2     Urpon Frisbert     State     CreditLimit       3     Hoos     Suite     State       4     Go Fahing Lidt     Unit 2     Harrow     Middlesex       5     Match Reint Tonnie, fisi kenner, Pl.     Roston     MA       5     Match Reint Tonnie, fisi kenner, Pl.     Submit Changes       Custhum     Order/Date     SalesRep       4     24     Carrier     Order/Date       5     Submit Changes     Custer	Image: Safety = Tools = Ref. *           Home         About         CustOrder           solution         togin
Application Name     Home     About       Custhum     Name     Address     City     State     CreditLimit       1     Lift Tours     276 North Drive     Burlington     MA     66700     •       2     Urpon Frisbert     Ratioolitu 33     Odo     Usime     27600     •       3     Hoops     Suite 415     Allenta     GA     75000     •       4     Ge Fahing Ltd     Uit 2     Harraw     Hiddless:     1100     •       Custhum     OrderNum     Carrier     Norde-Date     SalesRap	Home About CustOrder
Custhum     Name     Address     City     State     Credit/imit       1     Lift Tours     276 North Drive Burlington     MA     66700       2     Urpon Fridbett     Rattipoliku 33     Odlo     Usame     27600       3     Hoops     Suite 415     Allanta     GA     75000       4     G6 Fathing Lid     Ult 2     Harraw     Middless:     1000       5     Match Roint Transin, 66 Hammer PI     Boston     MA     11000       Custhum     Order/Num     Carrier     Order/Date     Sales/Bap       Custhum     Order/Num     Carrier     Order/Date     Sales/Bap	wittuma
Custhum         Name         Address         City         State         Credit/imit           1         Lift Tours         276 North Drive         Burlington         MA         66700         *           2         Urpon Fridsbeet         Rattipoliku 33         Oalo         Uusima         27800         *           3         Hoops         Suite 415         Atlanta         6A         75000         *           4         Ge Fahing Lid         Uit 2         Harrow         Middleasx         1500         *           Get Skies/Rep         Smith , Spike Louise         Forder MA         1100         *         *           Custhum         Order/Num         Carrier         Order/Date         Sales/Rep         *         *	editLimit 700 • • 800 • • 000 • • 000 • •
Cubrolum Famile     Andreas     City Statute     City Statute       1     Lift Tours     27 Month Drive Buildmin MA     65700       2     Urpon Frisbeet     Rottipoliu 33     Odio     Uusima     27600       3     Hoops     Suite 415     Atlanta     GA     75000       4     Ge Fahing Lid     Uit 2     Harrow     Middlesst     13000       5     Match Built Tzenic     66 Monner PL     Roston     MA     13000       Custitum     Order/Date     SalesRep     Suite 12/10:000 AMI 505	
2         Urpon Probect         Attoriation Virial         Submit Attack         Submit Attack </td <td></td>	
3         Hoops         Suite 415         Atlents         GA         75000           4         Go Fishing Lid         Unit 2         Harrow         Middlesex         15000         *           5         Motch Boint Teonic, 56 Homer PI         Rodran         MA         11000         *           Get SalesRep         Smith , Spike Louise         Submit Changes         Submit Changes           Custium         Order/Date         SelesRep         SelesRep           1         100:000 Atl 515         Standard Mail         2/2/1/98 12:00:000 Atl 515	000 000 000
4         Go Fahing Ltd         Unit 2         Harrow         Middlesex         \$5000           5         Match Boint Tennic, Ki Homer PI         Rodzin         Mid         11000         *           Get SalesRep         Smith , Spike Louise         Submit Changes         Submit Changes         Custium         OrderDate         SalesRep           Custium         OrderDate         SalesRep         Submit Changes         Submit Changes         Submit Changes	
S Match Point Tennis 66 Homer Pl Rodan MA 11000 * Get SalesRep Smith , Spike Louise Submit Changes Cuthum OrderNum Carrier OrderOate SalesRep 4 21 Standard Mail 2/21/98 12:00:00 AH SIS	200
Get SalesRep         Smith , Spike Louise         Submit Changes           CustNum         OrderNum         Carrier         OrderDate         SalesRep           4         21         Standard Mail         2/2/1/981 12:00:00 AMI SLS         SalesRep	
CustNum         Order/Num         Carrier         Order/Date         SalesRep           4         21         Standard Mail         2/2/1998 12:00:00 AM SLS         SLS	Submit Changes
4 21 Standard Mail 2/8/1998 12:00:00 AM 5LS	
4 6030 2/3/1998 12:00:00 AM BBB	

This example has shown you how to call an ABL procedure with any number of **INPUT** parameters, and exactly one non-entity **OUTPUT** parameter. As a review, and to allow you to copy code from this paper, here are the complete code elements used in this first Invoke example:

## First, the ABL procedure:

Next, the DomainService method:



Next, the click event handler for the new button in CustOrderPage.xaml.cs:



Next the completion event handler for the client event handler:



And finally, the one-line change to reset the **Text** property in the **SelectionChanged** event handler:



As I said at the outset, the first example is somewhat simplified, because it needs to return only a single scalar output parameter. In the second part of the paper I create a slightly more complex example to show how to return multiple output parameters as an object with two or more properties.

As before, I need to create a new stand-alone ABL procedure for a new service, which in this case goes beyond the previous one in that it takes the customer number as input, but also takes the **CreditLimit** from the client as an **INPUT-OUTPUT** parameter, and returns it, possibly modified, along with the **SalesRep** name. So in effect this procedure has two input parameters and two output parameters. I have also added a very simplistic bit of business logic here that says that if the **CreditLimit** hasn't been changed in the database, and if the customer **Balance** is less than half the **CreditLimit**, then the code increases the **CreditLimit** by twenty percent and returns that value:

```
/*_____
   File
            : GetRepAndCL.p
   Notes : Returns the SalesRep RepName and adjusts CreditLimit.
                                                              __*/
                                      AS INTEGER.
DEFINE INPUT
                 PARAMETER piCustNum
DEFINE INPUT-OUTPUT PARAMETER pdCreditLimit AS DECIMAL.
DEFINE OUTPUT
               PARAMETER pcSalesRepName AS CHARACTER.
FIND Customer WHERE Customer.CustNum = piCustNum.
IF (pdCreditLimit = Customer.CreditLimit) AND
(Customer.Balance < (Customer.CreditLimit / 2)) THEN
 pdCreditLimit = Customer.CreditLimit * 1.2.
FIND Salesrep WHERE Salesrep.SalesRep = Customer.SalesRep.
pcSalesRepName = Salesrep.RepName.
```

After saving the new procedure to the AppServerContent directory, I need to add it to the same proxy as the other procedures in the series. I add the procedure, and now I have four entry points in the proxy, each a stand-alone ABL procedure.

Let me take a moment to make sure it's clear how you would return the ABL **RETURN**-**VALUE** to the client if you needed to. You can right-click on an ABL .r file in ProxyGen's **Procedures** tab, and select **Customize...** 



Subapopert Procedure Toto Options Heb	Proxy Generator for Open Clients - C:\WebWorkspace\SilverlightSa	mple\CustOrderАpp.хpхg	
Customize Non-persistent Procedure in CustomerApp         Procedure To Run         Procedure To Run AppChect Urknown Setting         Procedure To Run AppChect To Run AppChect Before Image Setting	SubAppObject Procedure Tools Options Help		
ResumAEL RETURN VALUE Select the Return ABL RETURN-VALUE check box V Use AppOblect Unknown Setting V Use AppOblect Setting V Use AppOblect Setting V Use AppOblect Unknown Setting V Use AppOblect Set			
Procedure to Run AppSeverContent/UpdateCutDirdes::	Customize Non-persistent Procedure in CustOrderApp		
Procedure IR Rum AppServerContent/UpdateCut0Index: Method Name UpdatcCut0Index Return ABL RETURN-VALUE Select the Return ABL RETURN-VALUE check box V Use AppObject Unknown Seting Use AppObject Tempi size Seting	Procedure Parameters		
Method Name UpdaeCusthdes	Procedure to Run AppServerContent\UpdateCustOrders.r		
Petun ABL RETURN VALUE Select the Return ABL RETURN-VALUE check box V Use AppObject Urknown Setting Use AppObject Urknown Setting Use AppObject Temp-Table Setting	Method Name UpdateCustOrders	Cancel	
Return ABL RETURN VALUE Select the Return ABL RETURN-VALUE check box Use AppObject Urknown Setting Use AppObject Temp Table Setting	Description	Help	
Feature AEL RETURN-VALUE         Select the Return ABL RETURN-VALUE check box         Use AppObject Unknown Seting         Use AppObject Temp1 able Seting			
Select the Return ABL RETURN-VALUE check box	Return ABL RETURN-VALUE		
Select the Return ABL RETURN-VALUE check box			
Use AppObject Urknown Setting Use AppObject Temp Table Setting	Select the Return ABL RETURN-VALUE che	sk box	
Use AppOblect Temp-Table Setting Use AppOblect Temp-Table Setting Use AppOblect Temp-Table Setting	4		
	✓ Use AppObject Unknown Setting     ✓ Use AppObject Before	/e-Image Setting	
	<ul> <li>Over Appropriate verying</li> </ul>		

## The Customize dialog has an option to Return ABL RETURN-VALUE:

If you click that checkbox on, then the **RETURN-VALUE** is returned as a string with the name **retValue**, and you have to add that to the set of output parameters for the procedure. However, I'm not going to do that here.

When I click the **Generate** button, and then just **OK** the proxy properties dialog, I get a new version of the .NET proxy with four separate procedures defined.

Now I go back to the DomainService class I've been working with, to add support for the new procedure. As a reminder, here's the Invoke operation I added in the previous video.

[Invoke] public string GetSalesRep(int inputCustNum)
<pre>{     string outRepname;     _appObj.GetSalesRep(inputCustNum, out outRepname);     return outRepname; }</pre>

Note that because the underlying ABL procedure that is being invoked here through the proxy just returns a single output parameter, and no ABL **RETURN-VALUE**, the DomainService method can just define its return type as the return type of the ABL parameter – **CHARACTER** or **string** in this case – and simply return that one value.

In this second example, it's not going to be quite that simple. The ABL procedure returns two values, the INPUT-OUTPUT **CreditLimit** and the output **SalesRep** name. To represent that in the DomainService, I need to define a class with a property for each of those values. The class then becomes the return type for the other DomainService methods.

So before I define the new Invoke method, I define the class that encapsulates all its output parameters. There's a rule that all DomainService operations must use

serializable types for parameters and return types, so I add the **Serializable** attribute as an annotation in front of the parameter class:



The first property defined in the class represents the **CreditLimit**, the **INPUT-OUTPUT** parameter. The second one represents the **OUTPUT SalesRep** name. That's all that's needed in the class definition. Remember that because the call is asynchronous, the input parameters have already been passed to the request. Here I need to specify only the output parameters that come back to the completion event handler, including the output value for any input-output parameters.

Now I can define the class that invokes the ABL procedure through the proxy. Its return type is not a simple scalar type, but an object of the type I just defined. As I've done before, I give the method the same name as the ABL procedure, which is just my naming convention. There are two input parameters, the customer number and the current credit limit as held in the client:



Because my return type is now a class, I have to create an instance of the class to return. The first two parameters are accounted for as input to this method, but I still need to define a local variable to hold the third parameter when it comes back.

```
GetRepAndCLParams outParams = new GetRepAndCLParams();
string outRepname;
```

Next I invoke GetRepAndCL in the running AppObject instance. The first parameter is the customer number. The second is the credit limit, which is passed using the ref mode in C# to correspond to what ABL terms INPUT-OUTPUT. The third parameter is the output parameter.

try {			
-	_appObj.GetRepAndCL(inputCustNum, out outRepname);	ref	inoutCreditLimit,

Now I populate the parameter object I created, setting first the credit limit, in case that was modified on the server, and then the SalesRep name:

outParams.inoutCreditLimit = inoutCreditLimit;	
outParams.outRepname = outRepname;	
}	

The standard **catch** block completes the additions to the DomainService class:



As I've noted elsewhere, my exception handling code in these examples is very minimal, to keep the code brief, but even this simple catch block is useful because it will at least return an exception message to the client.

I can compile what I've just added here, and then do a Build again, to make sure everything gets regenerated properly. Once that's done, I can take another look at the generated DomainContext class to confirm what the user interface support code needs to pass to it. As I did before, I can look at the comments to check what I need to pass:



You can see that I just have to pass all the input parameters (including of course the input-output credit limit), followed by the callback method name, followed by a null for the user state. Remember that the call is asynchronous, so I pass all the inputs here and process the outputs later.

Having confirmed the signature of the call, I create a call to the DomainContext method in the user interface support code, **CustOrderPage.xaml.cs**. Starting with the click event handler for the Get SalesRep button, I can make some changes to run the new ABL procedure rather than the simpler one with the one **OUTPUT** parameter:

private void SalesRepButton\_Click(object sender, RoutedEventArgs e)
{
 CustOrder selectedCust = (CustOrder)CustomerGrid.SelectedItem;

First I add a simple sanity check here so that the code doesn't blow up if I click the button before I select a customer. (Be aware that my examples are very simplified in ways like this, so that I don't have to explain any more code than necessary.)

if (selectedCust != null)
{

Next I comment out the invoke operation for GetSalesRep, and in its place, I invoke the new one with two output parameters:



Note that the type of the invoke operation is now my new parameters class. In the DomainContext, I invoke the GetRepAndCL method that I just looked at. The input parameters come from the currently selected Customer in the grid, the CustNum and the CreditLimit. I name my event handler for the completion event OnSRB CL Completed, adding CL for CreditLimit.

Next I implement the new completion event handler. It's instructive to compare it with the simpler method, which as a reminder is shown again here:



First I add **CL** to the method name. Next I have to replace the **string** type with the parameter object type:



After the standard error check, instead of the **value** parameter of the invoke operation being a single value, it's now an object with a property of its own for each property in the parameters class. So I set the **Text** property of the SalesRep name TextBox to that property:



Next I need to re-display the CreditLimit value that's returned, in the grid. So I identify again the object that represents the currently selected row, the selectedItem. I set the CreditLimit cell in that row to the INPUT-OUTPUT value that was passed back from OpenEdge. Once again the Invoke operation's value parameter is an object, which also holds the CreditLimit property:



To review, when the user selects a Customer in the grid and clicks the Get SalesRep button, that click event fires the event handler in **CustOrderPage.xaml.cs**. The event handler runs the version of the **GetRepAndCL** method in the client's DomainContext class, passing the input parameters and the name of the completion method:



The DomainContext invokes the method that was just defined in the DomainService running on the Web server, asynchronously, which in turn invokes the ABL procedure via the AppObject proxy instance:



When the request completes, the new completion method OnSRB\_CL\_Completed runs, taking the invoke operation initiated by GetRepAndCL with its parameter object as input. The parameter object defines the output parameters, and the method copies the SalesRep name to its TextBox and the CreditLimit value back to the grid:



This example can reuse the Get SalesRep button and its TextBox, so there's no need to make any changes to the user interface itself.

If I run with the new methods compiled and and the project re-built, then when the test page comes up, I can select the CustOrder page, and Customer 2. I see the

Orders that are retrieved by the other user interface event, **SelectionChanged** of the Customer grid:

🖉 CustOrderi	Page Page - Wi	indows	Internet Explor	er						=D×
<del>6</del> 0•	http://local*	hast:528	78/CustOrderPro	jectTestPage.asp	x:#/CustOrde	rPage			💌 🔂 🐓 🗶 🎦 Google	ρ.
🔆 Favorites	👍 🄏 Sugge	ested Sib	es 👻 🙋 Web Sl	ice Gallery 💌						
CustOrder	Page Page								🏠 • 🔝 - 🖃 👼 • Page • Safety	r * Tools * 😥 * **
ar	plication N	Name							Home About	OustOrder
<b>•</b> •••	pircution	Turric						_		
_										login
CustNum	Name		Address	City	State	CreditLim	it in the second se			
1	Lift Tours		276 North Dri	ve Burlington	MA	66700		•	•	
2	Urpon Frisbe	set	Rattipolku 33	Oslo	Uusima	22600				
3	Hoops		Suite 415 Atlanta		GA	75000				
4	Go Fishing L	.td	Unit 2	Harrow Middlesex 1		15000	15000			
5	Match Point	Tennis	66 Homer Pl	Boston MA 11		11000		*	•	
Get	SalesRep					SL	bmit Changes			
CustNum	OrderNum	Carri	ier OrderDate			SalesRep				
2	94	Stan	dard Mail	10/4/1997 12:00:00 AM		DKP			•	
2	125	FlyBy	Night Courier	12/20/1997 1	2:00:00 AM	DKP				
2	171	Stan	dard Mail	11/30/1997 1	2:00:00 AM	DKP		-		

Now I undo the little changes I made to this row in the update presentation, removing the t from the end of the name, and changing the street number back to 3. Note the before value of the CreditLimit, 22600:

	lerPage Page - Wir	ndows Internet Explore	ir.			
0	<ul> <li>Image: Antipactic Action</li> <li>Antipactic Action</li> <li>Antipactic</li></ul>	ost:52878/CustOrderProje	ctTestPage.asp	∞#/CustOrde	rPage	💌 🔂 🍫 🗙 🚼 Google 🖉
Favorite	es 🛛 🏤 🏉 Sugger	sted Sites 🔹 🙋 Web Slic	e Gallery 🔻			
CustOrd	derPage Page					🏠 * 🔂 🗉 📾 🐐 Page * Safety * Tools * 😢
5	Application N	lame				Home About OustDerder
<u> </u>		anne				
						login
CustNun	m Name	Address	City	State	CreditLimit	
1	Lift Tours	276 North Driv	e Burlington	MA	66700	÷
2	Urpon Frisber	e Rattipolku 33	Oslo	Uusima	22600	
3	Hoops	Suite 415	Atlanta	GA	75000	
4	Go Fishing Lt	d Unit 2	Harrow	Middlesex	15000	
5	Match Point 1	Tennis 66 Homer Pl	Boston	MA	11000	
G	let SalesRep				Subr	it Changes
CustNun	m OrderNum	Carrier 0	OrderDate		SalesRep	
2	94	Standard Mail 1	0/4/1997 12:	MA 00:00	DKP	•
2	125	FlyByNight Courier 1	2/20/1997 12	2:00:00 AM	DKP	
2	171	Standard Mail 1	1/30/1997 12	2:00:00 AM	DKP	

Now I click the Get SalesRep button. This invokes the new GetRepAndCL method, and in turn the new ABL procedure of the same name, which not only retrieves the SalesRep name but updates the credit limit if it satisfies the business logic on the backend:

	prication N	ame	_	_	_			Home	About	login
CustNum	Name	Address	City	State	CreditLimit					
1	Lift Tours	276 North Driv	e Burlington	MA	66700		•			
2	Urpon Frisbe	e Rattipolku 3	Oslo	Uusima	27120		-			
3	Hoops	Suite 415	Atlanta	GA	75000					
4	Go Fishing Lt	d Unit 2	Harrow	Middlesex	15000					
5	Match Point	Cennis 66 Homer Pl	Boston	MA	11000		-			
Get !	SalesRep Pit	t , Dirk K.			Sub	mit Changes				
CustNum	OrderNum	Carrier	OrderDate		SalesRep	Challen a c	in Channel I.			
2	94	Standard Mail	0/4/1997 12:	00:00 AM	DKP	Circk the Subr	nit Changes button			
2	125	FlyByNight Courier	2/20/1997 13	2:00:00 AM	DKP	· · · · · · · · · · · · · · · · · · ·				
2	171	Standard Mail	1/30/1997 12	2:00:00 AM	DKP		-			
							•			

And after clicking the Get SalesRep button, the SalesRep name is displayed, and apparently the Balance for this Customer is less than half the CreditLimit, so the CreditLimit has been increased.

Now, the business logic didn't update the database, it just returned the new CreditLimit value to the client. If I now decide I want to save that new value, along with the other changes I made, I can press the Submit Changes button that was added to the application in the update example, and those changes are all sent back to the server in a change set and saved to the database. To verify that, I can get out of the CustOrder page, and then reselect it, and see that all the values for Customer 2 have been re-retrieved, confirming that they were saved to the database:

CustOrderP	age Page - Windows	Internet Explorer							=D×
ا • 💽 🕻	http://localhost:528	178/CustOrderProject	tTestPage.asp	oc#/CustOrde	rPage		•	🖸 🔂 🏘 🗙 🚼 Google	ρ.
Favorites	🍰 💋 Suggested Site	es 🔹 🙋 Web Slice	Gallery 🕶						
GustOrderP	age Page							🛐 * 🔝 - 🖃 📾 * Page * Safety *	Tools • 😧 • 👌
👌 Ap	plication Name							Home About	CustOrder
									login
CustNum	Name	Address	City	State	CreditLimit				
1	Lift Tours	276 North Drive	Burlington	MA	66700		*		
2	Urpon Frisbee	Rattipolku 3	Oslo	Uusima	27120		-		
3	Hoops	Suite 415	Atlanta	GA	75000				
4	Go Fishing Ltd	Unit 2	Harrow	Middlesex	15000				
5	Match Point Tennis	66 Homer Pl	Boston	MA	11000		*		
Get S	alesRep				Subm	it Changes			

If I select Customer 4, and click the Get SalesRep button, I get the SalesRep but no change to the CreditLimit. The grid doesn't display the Balance value, but if I superimpose the little FOR EACH Customer result shown below, you can see that, whereas the Balance for Customer 2 satisfies the criterion of being less than half the

CreditLimit, Customer 4 does not, so there was no change to the CreditLimit value that was passed back. A more responsive user interface that makes all this clearer is left as a Silverlight exercise for the reader.

	sge Page						🙆 • 6	) - 🖂 🖶 + Pag	e • Safety •	Tools + 🚺
App	plication Na	me						Home	About	CustOr
			_		_	ÖE Cust Num	Name	Balance	 Credit Limit	
ustNum	Name	Address	City	State	CreditLimit					
	Lift Tours	276 Nort	h Drive Burlingt	in MA	66700		Lift Tours	903.64	66,700	
	Urpon Frisbee	Rattipolk	u 3 Oslo	Uusima	27120	2	Ulpon Frisbee	437.63	27,120	
	Hoops	Suite 41	5 Atlanta	GA	75000	Å	Figure 1 td	14.235.14	15,000	
	Go Fishing Ltd	Unit 2	Harrow	Middlesex	15000	5	Match Point Tennis	0.00	11,000	
	Match Point Te	nnis 66 Home	r Pl Boston	MA	11000	6	Fanatical Athletes	1,202.66	38,900	
Cat St	alasBan Coult	th. Collee Louis	-		Submit Cha	7	Aerobics valine Ky	1,112.44	13,500	
OR 3	arearcep Jamin	an y aprice cours	•		Submit Chi	0	Game Set Match	8,254.00	15,000	
ustNum	OrderNum	Carrier	OrderDate	Sales	Rep	2	Pihtiputaan Pyora	1,242.14	29,900	
	21	Standard Mail	2/8/1998 12:00:	00 AM SLS						
	6030		2/3/1998 12:00:	DO AM BBB						•
						_				
						Procedu	re complete. Press space	bar to continue.		14

The complete code for all the pieces of this second example is repeated here. First, the ABL procedure:

/*
File : GetRepAndCL.p
Notes : Returns the SalesRep RepName and adjusts CreditLimit.
DEFINE INPUT PARAMETER piCustNum AS INTEGER. DEFINE INPUT-OUTPUT PARAMETER pdCreditLimit AS DECIMAL. DEFINE OUTPUT PARAMETER pcSalesRepName AS CHARACTER.
<pre>FIND Customer WHERE Customer.CustNum = piCustNum. IF (pdCreditLimit = Customer.CreditLimit) AND (Customer.Balance &lt; (Customer.CreditLimit / 2)) THEN pdCreditLimit = Customer.CreditLimit * 1.2.</pre>
FIND Salesrep WHERE Salesrep.SalesRep = Customer.SalesRep. pcSalesRepName = Salesrep.RepName.

Next, the parameter class that defines the output parameters in the DomainService class:



Next, the Invoke method in the DomainService class that invokes the ABL procedure through the proxy:



And finally, the modified click event handler in CustOrderPage.xaml.cs, and its completion event handler:





This brings me to the end of the series of presentations on using Silverlight with WCF RIA Services. I hope this information has been useful. Though there's much to learn about all the controls and capabilities that Silverlight supports, the basic thread of retrieving and updating data from an OpenEdge application via a set of proxies, entity class definitions, and DomaInService classes should be pretty clear to you. At this point the rest is up to you. Thanks for watching and reading.