

Index

NETConductor – Returning data 3
Adding related information to simple replies 3
Adding related information to string-list replies..... 4
Adding related information to record-set replies 5
Adding reply formatting 7
Conclusions..... 8

Documentation:

21/05/2009 Alfred B. Thordarson
Created document

NETConductor – Returning data

This document explains the different ways the <parent> construct can be used with the Return... operations of the NETConductor. In other words it describes how to return data from NAV in a hierarchical format.

Before you read this document you should read the NETConductor-ReferenceGuide.pdf and especially the NAV API functions that accept a Name as first parameter.

Adding related information to simple replies

The following Dynamics NAV code will, when called using the RunCodeUnit operations of the Basics web service of the NETConductor, return a single reply:

```
// Variables:
NETConductor [Codeunit 84920 NETConductor]
```

```
// C/AL Source Code:
NETConductor.ReturnString('Name', 'Alfred');
NETConductor.ReturnString('Address', 'Seilugrandi');
NETConductor.ReturnString('City', 'Reykjavik');
NETConductor.ReturnString('Country', 'Iceland');
```

And the resulting XML looks like this:

```
<ReplySet name="63360965921461005335">
  <Reply type="String" name="Name" value="Alfred"/>
  <Reply type="String" name="Address" value="Seilugrandi"/>
  <Reply type="String" name="City" value="Reykjavik"/>
  <Reply type="String" name="Country" value="Iceland"/>
</ReplySet>
```

However, I would like the address to be indented under the name, so I change the code to contain the parent name:

```
NETConductor.ReturnString('Name', 'Alfred');
NETConductor.ReturnString(' <Name>Address', 'Seilugrandi');
NETConductor.ReturnString(' <Name>City', 'Reykjavik');
NETConductor.ReturnString(' <Name>Country', 'Iceland');
```

And the XML will show up like this:

```
<ReplySet name="63360965921461005335">
  <Reply type="String" name="Name" value="Alfred">
    <Reply type="String" name="Address" value="Seilugrandi"/>
  </Reply>
  <Reply type="String" name="City" value="Reykjavik"/>
  <Reply type="String" name="Country" value="Iceland"/>
</ReplySet>
```

This is better, but what if I have two names like this:

```
NETConductor.ReturnString('Name', 'Alfred');
NETConductor.ReturnString(' <Name>Address', 'Seilugrandi');
NETConductor.ReturnString(' <Name>City', 'Reykjavik');
NETConductor.ReturnString(' <Name>Country', 'Iceland');
```

```
NETConductor.ReturnString('Name','Kris');
NETConductor.ReturnString('<Name>Address','Broadwalk');
NETConductor.ReturnString('<Name>City','NY');
NETConductor.ReturnString('<Name>Country','USA');
```

The XML returned will be like this:

```
<ReplySet name="63360966151254938937">
  <Reply type="String" name="Name" value="Alfred">
    <Reply type="String" name="Address" value="Seilugrandi"/>
    <Reply type="String" name="City" value="Reykjavik"/>
    <Reply type="String" name="Country" value="Iceland"/>
  </Reply>
  <Reply type="String" name="Name" value="Kris">
    <Reply type="String" name="Address" value="Broadwalk"/>
    <Reply type="String" name="City" value="NY"/>
    <Reply type="String" name="Country" value="USA"/>
  </Reply>
</ReplySet>
```

This works as well!? How does the NET-Conductor know which "Name" to use as parent when the Kris 'Name' has also been added? The answer is: The NET-Conductor always uses the LAST node that fulfills the parent constraint.

What if I want to add the zip-code to the City reply of Alfred. I would do this:

```
NETConductor.ReturnString('<v:Name=Alfred/City>Zip','IS-107');
```

Now "v:Name=Alfred" tells the NETConductor to look for a reply, named "Name" and containing the value "Alfred". When the NETConductor has found that node it should look under that node (indicated by the '/') for a reply named "City". When it has found the "City" node it will create a string reply named "Zip" containing the value "IS-107" and the resulting XML looks like this:

```
<ReplySet name="63360966151254938937">
  <Reply type="String" name="Name" value="Alfred">
    <Reply type="String" name="Address" value="Seilugrandi"/>
    <Reply type="String" name="City" value="Reykjavik">
      <Reply type="String" name="Zip" value="IS-107"/>
    </Reply>
    <Reply type="String" name="Country" value="Iceland"/>
  </Reply>
  <Reply type="String" name="Name" value="Kris">
    <Reply type="String" name="Address" value="Broadwalk"/>
    <Reply type="String" name="City" value="NY"/>
    <Reply type="String" name="Country" value="USA"/>
  </Reply>
</ReplySet>
```

Adding related information to string-list replies

The following code will return a set of key/values.

```
// Variables:
NETConductor [Codeunit 84920 NETConductor]
StringList [Record 84921 NETConductor String List + Temporary:yes]
```

```
// C/AL Source Code:
StringList.add('A','Alfred B. Thordarson');
StringList.add('K','Kris Martinskov');
NETConductor.ReturnStringlist('Employees',Stringlist);
```

And the XML reply will be like this:

```
<ReplySet name="63360970964177707743">
  <Reply type="Hashtable" name="Employees">
    <KeyValuePair key="A" value="Alfred B. Thordarson"/>
    <KeyValuePair key="K" value="Kris Martinskov"/>
  </Reply>
</ReplySet>
```

If I want to return the number of strings returned in the "Hashtable" then I can do that just like I have been doing above:

```
NETConductor.ReturnInteger('<Employees>Count',2);
```

Which results in XML looking like this:

```
<ReplySet name="63360970964177707743">
  <Reply type="Hashtable" name="Employees">
    <KeyValuePair key="A" value="Alfred B. Thordarson"/>
    <KeyValuePair key="K" value="Kris Martinskov"/>
    <Reply type="Integer" name="Count" value="2"/>
  </Reply>
</ReplySet>
```

But what if I would like to return Kris' address under his key/value node. In that case we would do something like this:

```
NETConductor.ReturnString('<Employees/k:Kris>Address','Broadwalk');
NETConductor.ReturnString('<Employees/k:Kris>City','NY');
NETConductor.ReturnString('<Employees/k:Kris>Country','USA');
```

For each of these commands the NETConductor will first look for a reply named "Employees" and under that look for a "KeyValuePair" with "key" equal to "Kris", this is indicated with the k:Kris. The resulting XML will therefore be this:

```
<ReplySet name="63360971435984168550">
  <Reply type="Hashtable" name="Employees">
    <KeyValuePair key="Alfred" value="Alfred B. Thordarson"/>
    <KeyValuePair key="Kris" value="Kris Martinskov">
      <Reply type="String" name="Address" value="Broadwalk"/>
      <Reply type="String" name="City" value="NY"/>
      <Reply type="String" name="Country" value="USA"/>
    </KeyValuePair>
    <Reply type="Integer" name="Count" value="2"/>
  </Reply>
</ReplySet>
```

Adding related information to record-set replies

The following code will return a set of records.

```
// Variables:
NETConductor [Codeunit 84920 NETConductor]
CustRec [Record 18 Customer]
Recs [RecordRef]
Cols [Record 84920 NETConductor Field List + Temporary:yes]
```

```
// C/AL Source Code:
Recs.GETTABLE(CustRec);
Cols.add(CustRec.FIELDNO("No. "));
Cols.add(CustRec.FIELDNO(Name));
```

```
Cols.add(CustRec.FIELDNO(Balance));
NETConductor.ReturnRecordset('Customers',Recs,Cols);
```

This code will return an XML like this:

```
<ReplySet name="6336096378961329335">
  <Reply type="RecordSet" name="Customers">
    <Customer TableNumber="18" Caption="Customer">
      <No_ FieldNumber="1" Caption="No.">01445544</No_>
      <Name FieldNumber="2" Caption="Name">Progressive Home Furnishings</Name>

      <Balance FieldNumber="58" Caption="Balance">2310.38</Balance>
    </Customer>
    <Customer TableNumber="18" Caption="Customer">
      <No_ FieldNumber="1" Caption="No.">01454545</No_>
      <Name FieldNumber="2" Caption="Name">New Concepts Furniture</Name>
      <Balance FieldNumber="58" Caption="Balance">342529.44</Balance>
    </Customer>
    ...
  </Reply>
</ReplySet>
```

Now lets say I want to return some additional information on Customer 01445544. I would do it like this, using the \$-sign to indicate a fieldname:

```
NETConductor.ReturnDecimal('<Customers/$No_=01445544>ExtraInfo',-2010.38);
```

Or using the 'f:' prefix to indicate a field number:

```
NETConductor.ReturnDecimal('<Customers/f:1=01445544>ExtraInfo',-2010.38);
```

and the resulting XML will now be:

```
<ReplySet name="6336096378961329335">
  <Reply type="RecordSet" name="Customers">
    <Customer TableNumber="18" Caption="Customer">
      <No_ FieldNumber="1" Caption="No.">01445544</No_>
      <Name FieldNumber="2" Caption="Name">Progressive Home Furnishings</Name>

      <Balance FieldNumber="58" Caption="Balance">2310.38</Balance>
      <Reply type="Decimal" name="ExtraInfo" value="-2010.38"/>
    </Customer>
    <Customer TableNumber="18" Caption="Customer">
      <No_ FieldNumber="1" Caption="No.">01454545</No_>
      <Name FieldNumber="2" Caption="Name">New Concepts Furniture</Name>
      <Balance FieldNumber="58" Caption="Balance">342529.44</Balance>
    </Customer>
    ...
  </Reply>
</ReplySet>
```

Then you can add addition information to the last 'ExtraInfo' reply within the result, like this:

```
NETConductor.ReturnDecimal('<ExtraInfo>Thousands',2);
NETConductor.ReturnDecimal('<ExtraInfo>Hundreds',0);
NETConductor.ReturnDecimal('<ExtraInfo>Tens',1);
NETConductor.ReturnDecimal('<ExtraInfo>Units',0.38);
```

And the resulting XML will look like this:

```
<ReplySet name="6336096378961329335">
  <Reply type="RecordSet" name="Customers">
    <Customer TableNumber="18" Caption="Customer">
      <No_ FieldNumber="1" Caption="No.">01445544</No_>
```

```

    <Name FieldNumber="2" Caption="Name">Progressive Home Furnishings</Name>

    <Balance FieldNumber="58" Caption="Balance">2310.38</Balance>
    <Reply type="Decimal" name="ExtraInfo" value="-2010.38">
      <Reply type="Decimal" name="Thousands" value="2"/>
      <Reply type="Decimal" name="Hundreds" value="3"/>
      <Reply type="Decimal" name="Tens" value="1"/>
      <Reply type="Decimal" name="Units" value="0.38"/>
    </Reply>
  </Customer>
  <Customer TableNumber="18" Caption="Customer">
    <No_ FieldNumber="1" Caption="No.">01454545</No_>
    <Name FieldNumber="2" Caption="Name">New Concepts Furniture</Name>
    <Balance FieldNumber="58" Caption="Balance">342529.44</Balance>
  </Customer>
  ...
</Reply>
</ReplySet>

```

Adding reply formatting

Using the above functionality it should be possible to indent the information from Dynamics NAV as much as required

For further formatting of the replies from Dynamics NAV, using the NETConductor we would like to suggest that you use an XSLT and write your code like this:

```
NETConductor.UseXSLTMapper('Customers.xslt');
```

Your 'Customers.xslt' file should be located in your C:\inetpub\wwwroot\NETConductor\XSLT directory and might look like something like this:

```

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" encoding="iso-8859-1" />

  <xsl:template match="ReplySet">
    <kunnar>
      <xsl:apply-templates/>
    </kunnar>
  </xsl:template>

  <xsl:template match="Customer">
    <kunni kenni="{No_/text()}">
      <nafn><xsl:value-of select="Name/text()"/></nafn>
      <heimili>
        <heimilisfang><xsl:value-of
of select="Address/text()"/></heimilisfang>
        <heimilisfang2><xsl:value-of
of select="Address_2/text()"/></heimilisfang2>
        <borg><xsl:value-of select="City/text()"/></borg>
        <postkodi><xsl:value-of select="Post_Code/text()"/></postkodi>
        <land><xsl:value-of select="Country_Code/text()"/></land>
      </heimili>
    </kunni>
  </xsl:template>
</xsl:stylesheet>

```

Which would result in the reply looking like this:

```

<kunnar>
  <kunni kenni="32656565">
    <nafn>Antarcticcopy</nafn>
    <heimili>

```

```

    <heimilisfang>Katwilgweg 274</heimilisfang>
    <heimilisfang2> </heimilisfang2>
    <borg>Antwerpen</borg>
    <postkodi>BE-2050</postkodi>
    <land>BE</land>
  </heimili>
</kunni>
<kunni kenni="44180220">
  <nafn>Afrifield Corporation</nafn>
  <heimili>
    <heimilisfang>100 Maidstone Ave.</heimilisfang>
    <heimilisfang2> </heimilisfang2>
    <borg>Maidstone</borg>
    <postkodi>ME5 6RL</postkodi>
    <land>GB</land>
  </heimili>
</kunni>
  ...
</kunnar>

```

Conclusions

From the techniques described in this tutorial you should be able to provide any kind of XML from Dynamics NAV, using the NET·Conductor product from Rue de Net.



Vesturgata 3, 101 Reykjavik, Iceland

ruedenet@ruedenet.com

Tel: (+354) 414 5050

Fax: (+354) 414 5051