FusionCharts

Overview

FusionCharts are general-purpose charts, such as column, bar, line, area, and pie. Chart data can be supplied in a few different ways, depending on your needs. The chart can use statically defined data, or the data can load dynamically from an RPG program or database.

Charts can be used both in the Genie environment as well as with Rich Display Files. However, many chart options are unsupported in a Genie-only environment. Read about Genie specific charts in the separate, Genie Charts page.

Field Binding Dialog (Rich UI Only)

If you are using the Rich UI, then you can bind any of the chart properties to variables in an RPG program. For more information about binding, please visit the Field Binding page.

Chart Properties

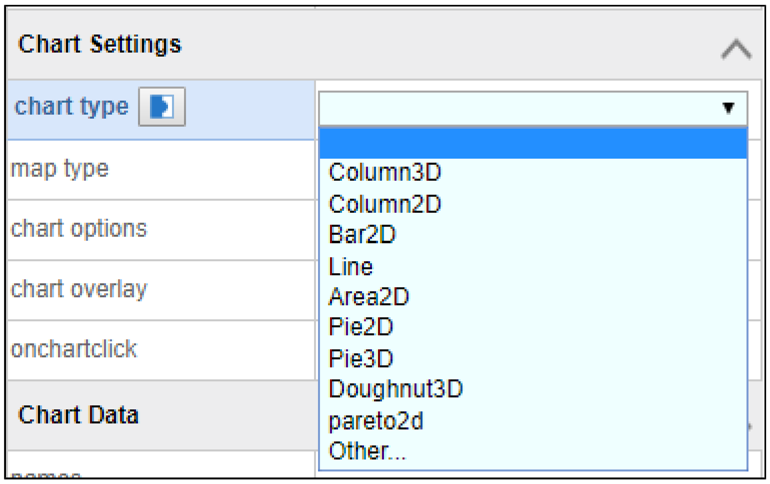
Properties specific to chart widgets are described below.

Identification

chart response - Specifies a response field to be populated with the name of the data-point selected by the user.

Chart Settings

***chart* type** - Specify the type of chart with this property. The most common types included in the designer's drop-down list are: Column3D, Column2D, Bar2D, Line, Area2D, Pie2D, Pie3D, and Doughnut3D.



There are also dozens of other chart types available. To specify a different type, click on "Other..." in the drop-down. The list becomes a text-box, and you can enter another chart type.

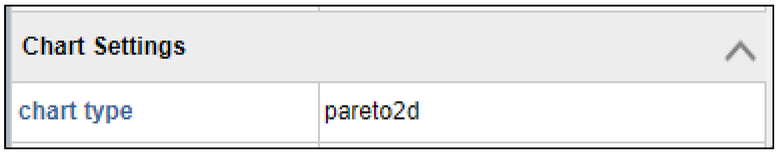


Chart Settings	
chart type	pareto2d

All available types are listed in the FusionCharts List of Charts.

Notes:

- in Visual Designer, a widget with type "Other" appears as a 2D bar chart, but outside of designer the desired chart is rendered correctly.
- 3D chart types can be much slower to render than 2D chart types. A page may take a few seconds to load with 3D types. Disabling animation can load the chart faster. See the example below in "chart options", for disabling animation.

map type - This property must be blank for charts.

chart options - Specifies chart options as a set of XML attributes that will be placed inside the FusionCharts <chart> tag. All attributes are found on the FusionCharts Attribute References. See also the Chart Attributes section below.

As a simple example, you may disable chart animation by including the following property in chart options:

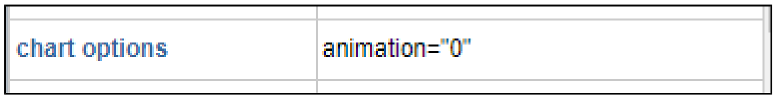


chart options	animation="0"

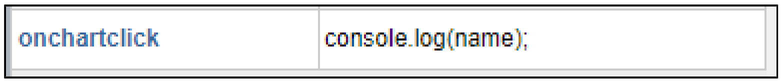
Notes:

- Animation makes charts fancier but causes pages to load more slowly. Also, because the chart is loaded first, slow charts cause other elements on your page to load slowly.
- If you are using RPG to generate the chart XML or JSON, then this field is ignored.

chart overlay - If the property is set to true, then the chart background color will depend on the chart type. Otherwise, the chart background color will be transparent. Read further in the FusionCharts setTransparent method.

onchartclick - Specify javascript to handle the chart-click event. When the user clicks the chart, then an event is generated with a parameter named, "name", whose value will be the name of the region clicked on the chart. There are a few ways to handle the event.

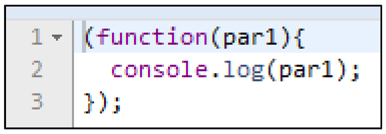
1. You may specify inline javascript code:



onchartclick	console.log(name);

With inline code, a variable named "name" is automatically defined and set to the name of the data column/bar/point that was clicked.

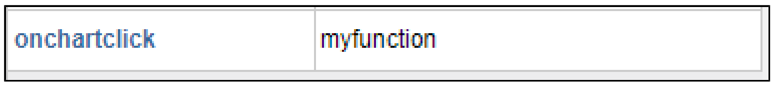
2. Create an anonymous function:



```
(function(par1){
  console.log(par1);
});
```

When using a function to handle the click, the first parameter is assigned with the name of the data column/bar/point that was clicked.

3. Specify a user-defined function:



onchartclick	myfunction

When using a function to handle the click, the first parameter is assigned with the name of the data column/bar/point that was clicked. User defined functions can be defined in Custom External Javascript.

Chart Data

One way to define the data presented on the chart is by setting the names and values from within Visual Designer. Defining data this way, the *names* and *val ues* fields are comma-separated lists, and the lists must be the same length. Note, however, that **only single-series** charts are supported when chart data is supplied this way.

names - Specify the names of the columns/bars/points plotted in the chart. This is the independent variable.

values - Specify the value for a column/bar/point. This is the dependent variable.

2D Column Chart Example:

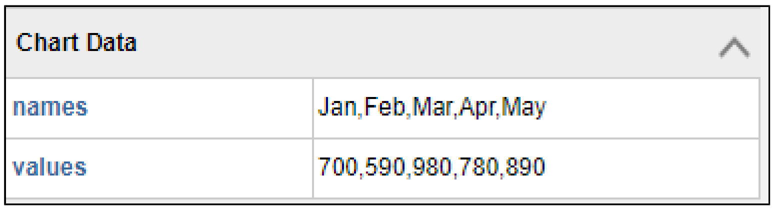
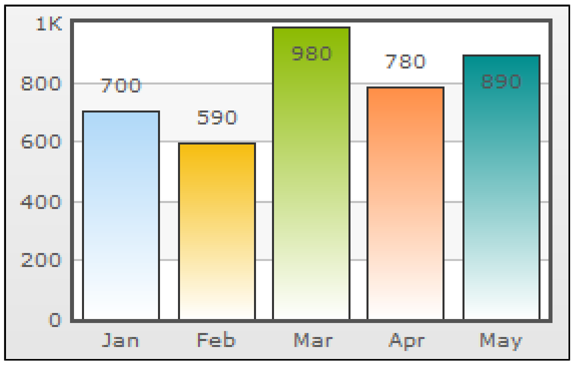


Chart Data

names	Jan,Feb,Mar,Apr,May
values	700,590,980,780,890

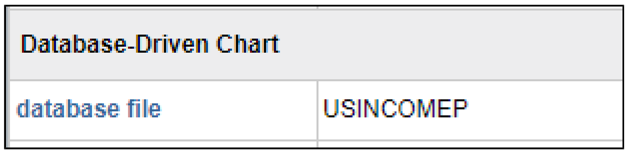


Hint: it is possible to create a dynamic chart by binding the *names* and *values* properties to your RPG program. However, other methods may better suit your needs.

Database-Driven Chart

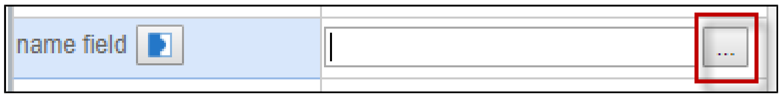
Another method to generate dynamic charts is by specifying a database file to drive the data. This is a fairly simple way to connect a chart widget to a database–no XML or JSON need be generated by your RPG program. However, database-driven charts can only display a **single data series**. (For multi-series charts, see the Dynamic Chart section below.)

To start, enter an existing data file into the database file property:

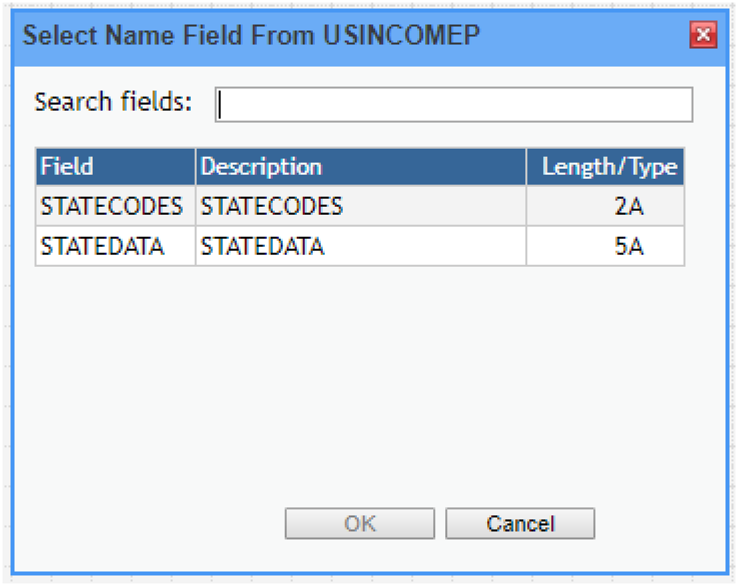


Database-Driven Chart	
database file	USINCOMEP

Next, specify a column for the name field. Click on the ellipses to display a helper dialog box:



If the database file is found in your library list, then a helpful dialog box appears:

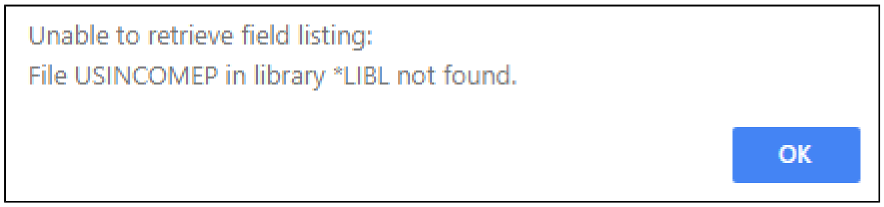


Select Name Field From USINCOMEP

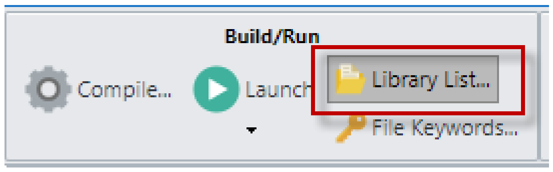
Search fields:

Field	Description	Length/Type
STATECODES	STATECODES	2A
STATEDATA	STATEDATA	5A

If instead of a helpful dialog box you get an error message like this,

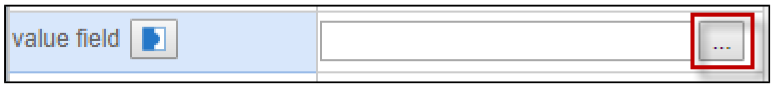

Unable to retrieve field listing:
File USINCOMEP in library *LIBL not found.

then you may need to modify the Library List by clicking the "Library List..." button:

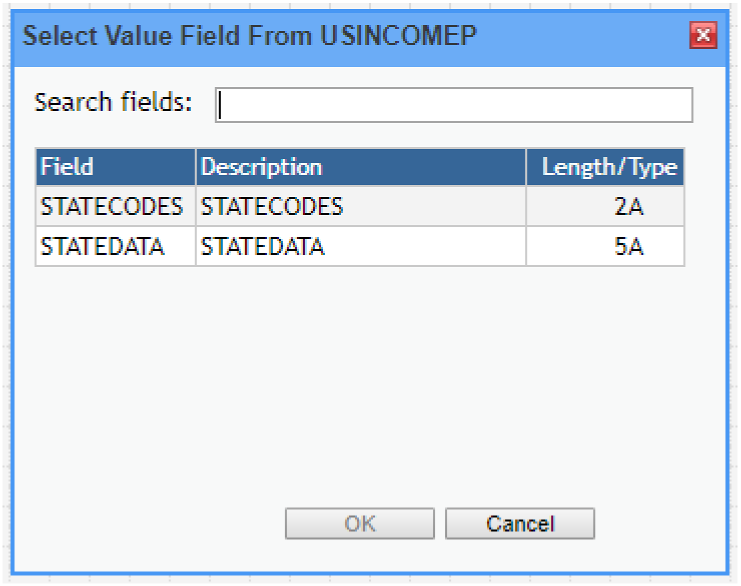


Once you see the helpful dialog box, choose a column to be used for the name field. Data from this column becomes the independent variable in the chart; e.g. for a bar chart the data becomes the bar names.

Next, choose a database column to supply the chart values. Click on the ellipses icon in the value field property



You'll see a helpful dialog box for choosing the Value field, and you can pick one:

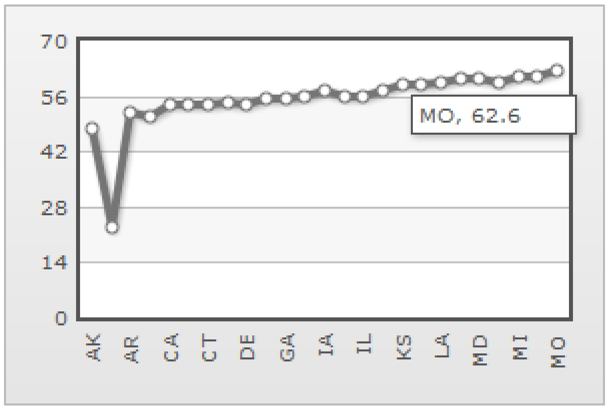

Select Value Field From USINCOMEP

Search fields:

Field	Description	Length/Type
STATECODES	STATECODES	2A
STATEDATA	STATEDATA	5A

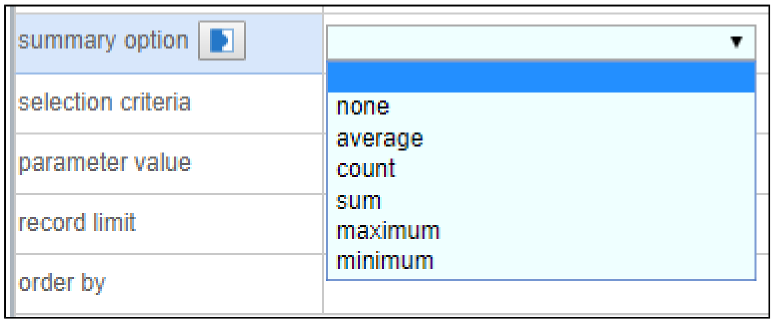
Choose a column and click OK.

For basic data, those three properties–database file, name field, value field--are the minimum required settings to use the Database-Driven chart feature. Below is a 2D line chart driven by a sample database file:



For more complicated database files, you may need to specify other chart properties, as explained below:

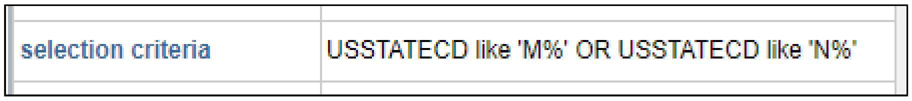
summary option - Determines how values are used when creating the chart.



- **none:** No aggregation will be done. Same as leaving the field empty.
- **average:** Finds the average dependent value corresponding to each ***name field*** value.
- **count:** Counts the number of duplicate ***name field*** values within your data (ignores the ***value field***).
- **sum:** Shows the sum of the dependent values for each ***name field*** value.
- **maximum:** Compares maximum dependent values among ***name field*** values.
- **minimum:** Compares minimum dependent values among ***name field*** values

When summary option is used the data is sorted by the "name field".

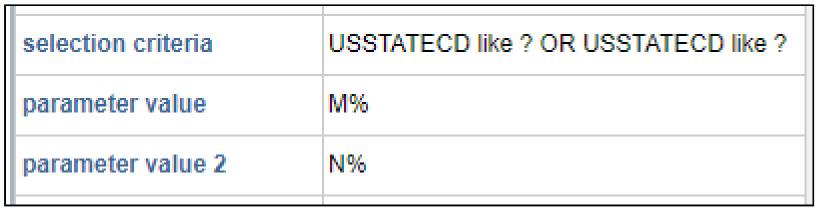
selection criteria - You can limit the query results by adding SQL expressions to this property:



selection criteria	USSTATECD like 'M%' OR USSTATECD like 'N%'

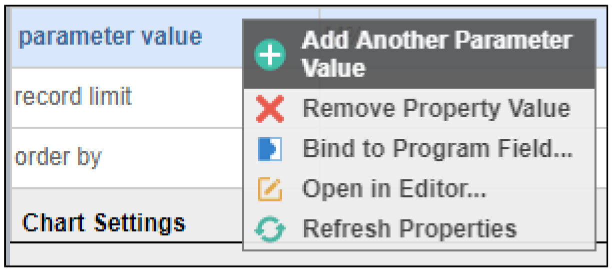
You can also use parameter markers in the expression in conjunction with the *parameter value* property. Specify a marker using a question mark.

parameter value - Specify a value for a parameter marker used in "selection criteria" property. Profound UI will accept any parameter marker values which are not bound to program fields. Parameter markers are numbered in order of occurrence, from left to right



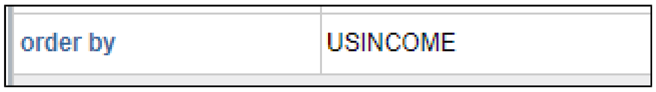
selection criteria	USSTATECD like ? OR USSTATECD like ?
parameter value	M%
parameter value 2	N%

To specify multiple parameter marker values, right-click the "parameter value" property and select Add Another Parameter Value:



record limit - Specify a limit on how many records are used in the chart; e.g. 5.

order by - Specify which fields determine the order of the items. This property is ignored when "summary option" is used.



order by	USINCOME

Click the ellipses to use a helper dialog box for choosing a column.

Dynamic Chart

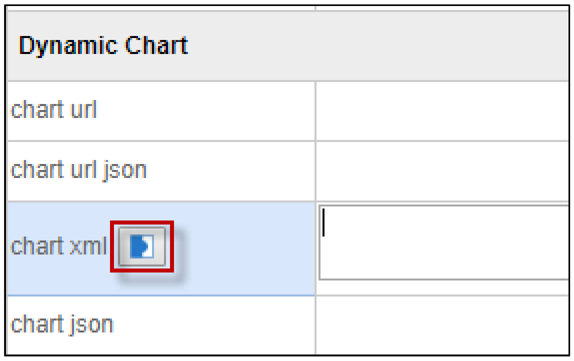
Lastly, there is another method to define chart data for Rich Display Files: by binding either the XML or JSON properties to an RPG program. This is much more complicated that the other methods, but it enables you to use chart features that the other method cannot support–for instance, Multi-Series charts.

See Field Binding for details about binding any property in a Rich Display file. To generate dynamic data, your RPG program can generate the XML or JSON text and put the result in the bound variable. Use either XML or JSON, but not both.

chart xml - Specify the raw XML to be used in the chart. (You can see example XML at this FusionCharts page. Find a sample chart and click on the XML tab below the chart.)

Below is a simple example of binding chart XML so you can generate a dynamic chart.

Click on the text "chart xml" to display the Binding button:



Dynamic Chart	
chart url	
chart url json	
chart xml	
chart json	

Click the Bind button to display a binding dialog:

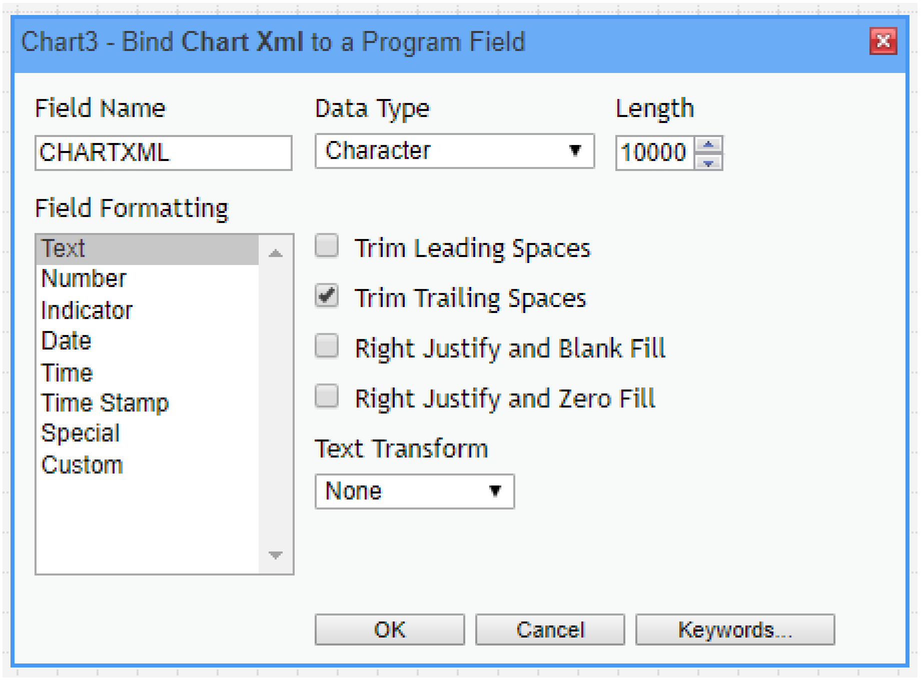


Chart3 - Bind Chart Xml to a Program Field

Field Name: CHARTXML
Data Type: Character
Length: 10000

Field Formatting: Text (selected), Number, Indicator, Date, Time, Time Stamp, Special, Custom

☐ Trim Leading Spaces
☑ Trim Trailing Spaces
☐ Right Justify and Blank Fill
☐ Right Justify and Zero Fill

Text Transform: None

Put in a field name that the RPG program will write to; e.g. CHARTXML. Use Character data type and use a length large enough to hold the XML you will generate; e.g. 10000.

Press OK, and you should see the chart xml property is bound to CHARTXML. (The green text indicates that the property is bound.)

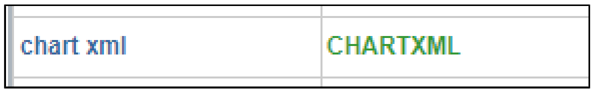


chart xml	CHARTXML

With that property bound, your RPG program can write to the CHARTXML field, and that data will be used to generate the chart widget when the page loads.

chart json - Specify the JSON data used to generate the chart. This is conceptually the same as *chart xml*. Use the example above as a guide, and bind to *c hart json* instead. Perhaps use a different field name, such as CHARTJSON.

You can see example JSON at this FusionCharts page. Find a sample chart, and its corresponding JSON code is below.

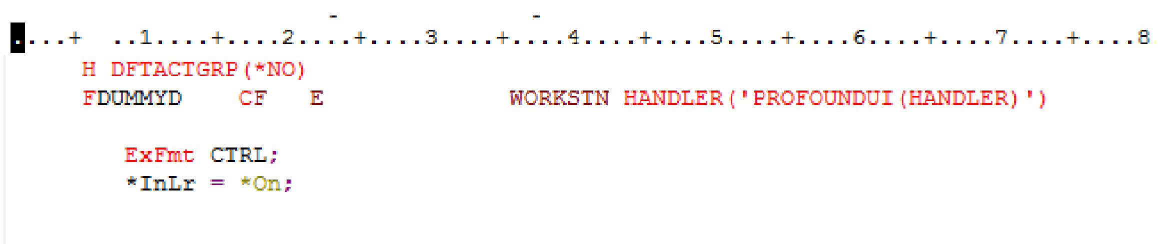
chart url - This property provides another way to generate a charts. Specify the URL of a web-service that returns chart data as XML text.

chart url json - Specify the URL of a web-service that returns chart data as JSON text. (Available with Profound UI Version 5, Fix Pack 7.0 and later.)

For any of the four properties under the Dynamic Chart category, you must only use one. Furthermore, these methods should not be used in conjunction with the Chart Data properties (i.e. names/values), nor should Database-Driven chart be used with these properties.

Limitations

Normally you can preview a design by clicking *Launch > Launch Preview...*, and a preview will open in a new tab. With Database-Driven charts, however, the preview cannot connect to a database file because it doesn't know the library. You will need to save and compile the display file, and then create an RPG program to drive the page. Your program may be as simple as this:



```
 ...+  ..1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     H DFTACTGRP(*NO)
     FDUMMYD    CF   E             WORKSTN HANDLER('PROFOUNDUI(HANDLER)')

        ExFmt CTRL;
        *InLr = *On;
```