



BLUEPHOENIX  
**ASNA**

---

**DataGate for .NET Bidirectional Text Support**

## Contents

<b>DataGate for .NET Bidirectional Text Support.....</b>	<b>1</b>
<b>Alternate Server Text Translation Interface.....</b>	<b>1</b>
<b>Bidi Text Translation.....</b>	<b>1</b>
<b>Configuring the iSeries for Bidi-enabled DataGate Connections .....</b>	<b>2</b>
<b>Configuring DataGate for Bidi-enabled Connections .....</b>	<b>3</b>
<b>Appendix A – Implementing and Configuring an Alternate Server Text Translation Provider .....</b>	<b>7</b>

## DataGate for .NET Bidirectional Text Support

RND is releasing an enhancement to DataGate support for iSeries databases. This enhancement involves text stored on the iSeries containing characters normally read from right-to-left (unlike English text, which is normally read from left-to-right, like the text in this sentence). This support will only be accessible through the .NET DataGate client.

---

### Alternate Server Text Translation Interface

This release of DataGate for .NET will feature a new library interface, which is designed to allow ASNA and its users create custom server text translators, by implementing subclasses of the .NET Framework's **System.Text.Encoding** class and associated classes. Currently, DataGate relies on the database server to provide a reasonable translation of character fields to and from Unicode. In extraordinary cases, such as iSeries-based bidirectional text, the database server is not able to evoke reliable translations. For these cases, the server text translation interfaces provide an alternative, client-based, plug-in capability to replace the standard DataGate Unicode translation.

These custom text translators will be loaded, via application configuration, by the **ASNA.DataGate.Client** library. Users will be able to associate a text encoding name, and other custom translator settings, with a registered "database name", or SourceProfile object. DataGate Studio will support the configuration of the custom translator in the "Advanced" connections properties dialog of the DataGate Explorer tool. **Appendix A** explains the interface in full detail.

---

### Bidi Text Translation

Right-to-left (RTL) text is most often seen in Middle Eastern languages, such as Arabic and Hebrew. Over the years, iSeries users and IBM have adopted conventions for entering and rendering text data, which collectively are inconsistent and complex to manage in the .NET era of Unicode character processing. Particularly difficult for current DataGate users are iSeries database fields containing a mix of RTL and left-to-right (LTR) text. This sort of mixed RTL/LTR text is referred to as bidirectional text, or simply, "bidi" text.

The DataGate bidi support enhancement consists of the server text translation interface, to be implemented by a bidi "provider." The provider is loaded by the DataGate run-time assembly to translate iSeries EBCDIC text fields to and from Unicode text. The provider will allow bidi text to be used and rendered in DataGate and Visual RPG programs. The character translators implemented by the provider will be configurable, via traditional DataGate "database name" configuration, to specify EBCDIC bidi conventions such as "String Type", and implicit/visual-style transformations.

As part of the enhancement, ASNA will ship its own bidi provider assembly, which will include Unicode transformations for the most common iSeries bidi CCSIDs, as follows:

CCSID	BidiEncoding Name
1255	EBCDIC-CP1255
1256	EBCDIC-CP1256
420	EBCDIC-CP420
424	EBCDIC-CP424
856	EBCDIC-CP856
862	EBCDIC-CP862
864	EBCDIC-CP864
1089	EBCDIC-ISO_8859-6
916	EBCDIC-ISO_8859-8

**Table 1. CCSIDs supported by the ASNA bidi provider.**

For the above CCSIDs, all string types currently supported by the iSeries will be supported by the ASNA bidi provider. Additionally, advanced users are able, and are encouraged to implement and/or use an alternate bidi provider for other bidi CCSIDs and special data entry conventions. Please see Appendix B for information on implementing and configuring a bidi provider assembly.

*Note that this DataGate bidi support is offered only for the DataGate .NET client. Also, the only tool currently available for configuring a bidi-enabled client connection is DataGate Studio.*

---

## Configuring the iSeries for Bidi-enabled DataGate Connections

By default, DataGate connections are not be enabled for bidi support. To enable bidi text transformations, you must select an appropriate bidi “encoding”, based on the text to be accessed on the iSeries. On the iSeries, the following factors should influence your selection:

- The CCSID of the file data to be accessed.
- The CCSID of the DataGate/400 job.

Note that on the iSeries, the database implicitly converts character data from the CCSID in which it is stored to the CCSID of the job. If the CCSIDs are the same, or the job CCSID is the special CCSID 65535, no implicit conversion occurs. DataGate/400 cannot control this conversion; therefore **it is the responsibility of the user to ensure that the implicit conversion is valid.**

Usually, the validity of the database's implicit conversion can be verified by using a "greenscreen" iSeries session to access and render the data. If the data is rendered correctly in the greenscreen, you should use the session's CCSID in the DataGate/400 connection job, unless the session's CCSID is 65535. If the session job's CCSID is 65535, use the CCSID of the stored data for the DataGate/400 connection job.

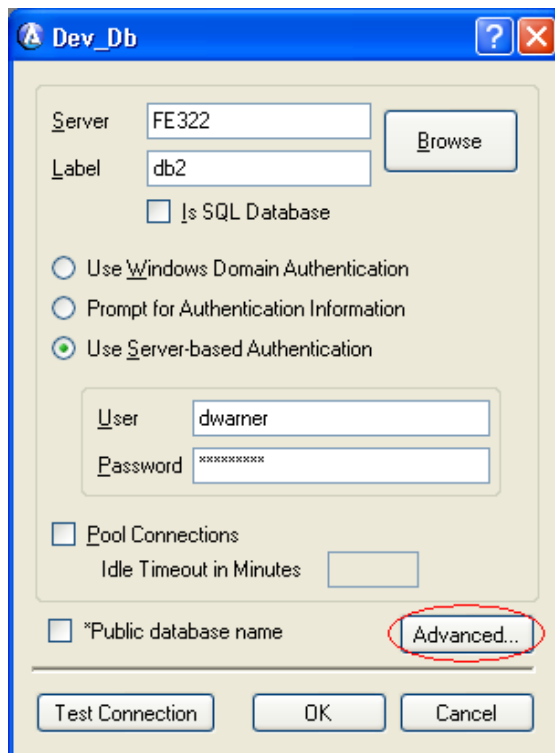
There are many ways to configure the CCSID of a DataGate/400 job, and enumerating them all is beyond scope here. However, the most common way to specify a DataGate job CCSID is to set the CCSID of the user profile specified in the DataGate database name. The user profile CCSID can be set with the iSeries **CHGUSRPRF** command.

Make note of your selection for the DataGate job CCSID and user profile, since these will be used to configure the bidi encoding of the client connection with DataGate Explorer.

---

## Configuring DataGate for Bidi-enabled Connections

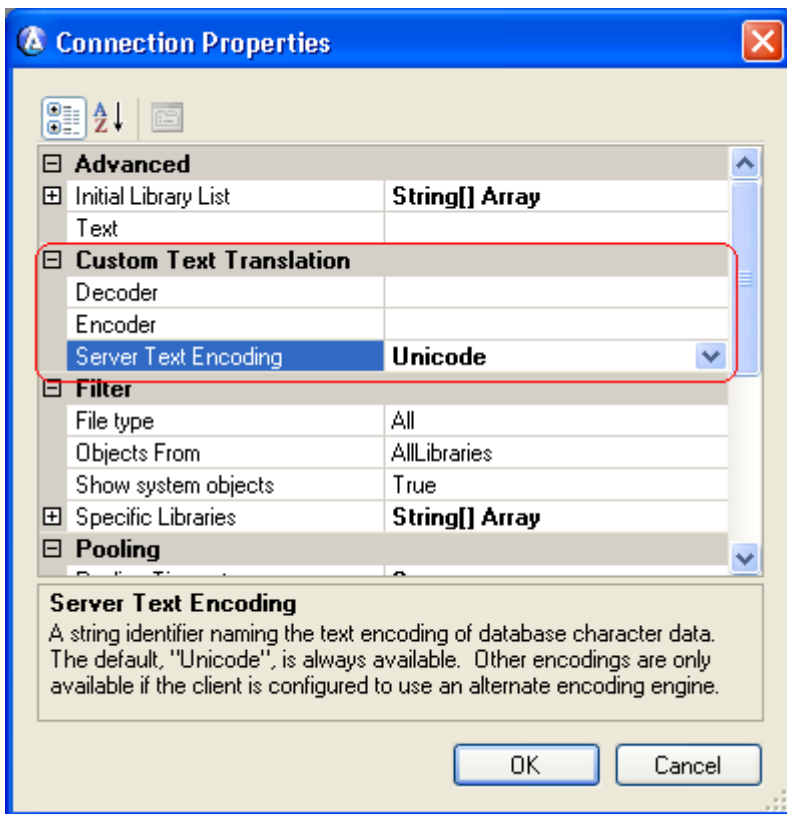
To enable the client for bidi support, you must configure the DataGate connection profile, using the source profile configuration dialog window in DataGate Studio. This dialog, shown in Figure 1, is accessible via the DataGate Explorer tool, with the **Modify Connection** command on a Connections node, or the **Edit Properties** command on a Database Names node.



*Figure 1. Click the Advanced button on the source profile configuration dialog to access bidi configuration properties.*

Before updating the bidi properties, take note of the value of the User on the dialog in Figure 1. In this example, the user “dwarner” is selected as the iSeries user profile configured to use a bidi CCSID. Here we assume that this profile will be used to configure the DataGate/400 job CCSID (as outlined in the previous section).

The bidi configuration properties for a connection are accessed by clicking the “Advanced...” button on the dialog, as circled in red in Figure 1. This will display a second dialog, containing a property grid control which includes the bidi configuration properties, as shown in Figure 2.



*Figure 2. By default, no alternate text conversion is configured, as shown by the "Unicode" property value for the "Server Text Encoding" property.*

To select the text encoding representing the bidi CCSID of the DataGate/400 job, click the drop-down button on the cell to the right of the Server Text Encoding property. This will display the encodings available from the provider assembly. The ASNA bidi provider will provide the encodings listed in Table 1, and these will be shown in the drop-down list. Select the encoding name from Table 1 that matches the DataGate/400 job CCSID shown in the table's CCSID column.

As an example, suppose that the “dwarner” user profile selected in Figure 1 is configured to use CCSID 424. We would therefore select “EBCDIC-CP424” for the Server Text Encoding property, as shown in Figure 3.

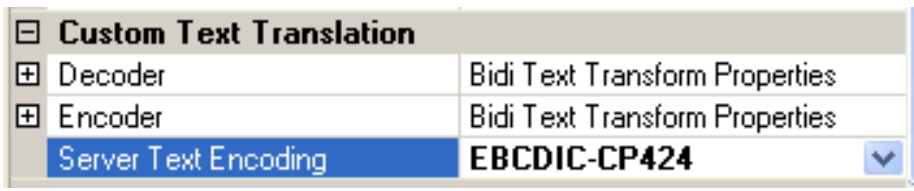


Figure 3. Selecting a valid encoding name will enable the Decoder and Encoder properties.

After selecting the encoding, you should review and update the properties effecting the interpretation of bidi data stored in the database. The Decoder and Encoder fields contain these properties. Expand the properties by clicking the “+” widget to the left of the fields. The Decoder properties effect the translation from EBCDIC to Unicode (that is, the translation of text stored in the database to Unicode as used in DataGate programs). Encoder properties effect translations in the opposite direction: Unicode to EBCDIC.

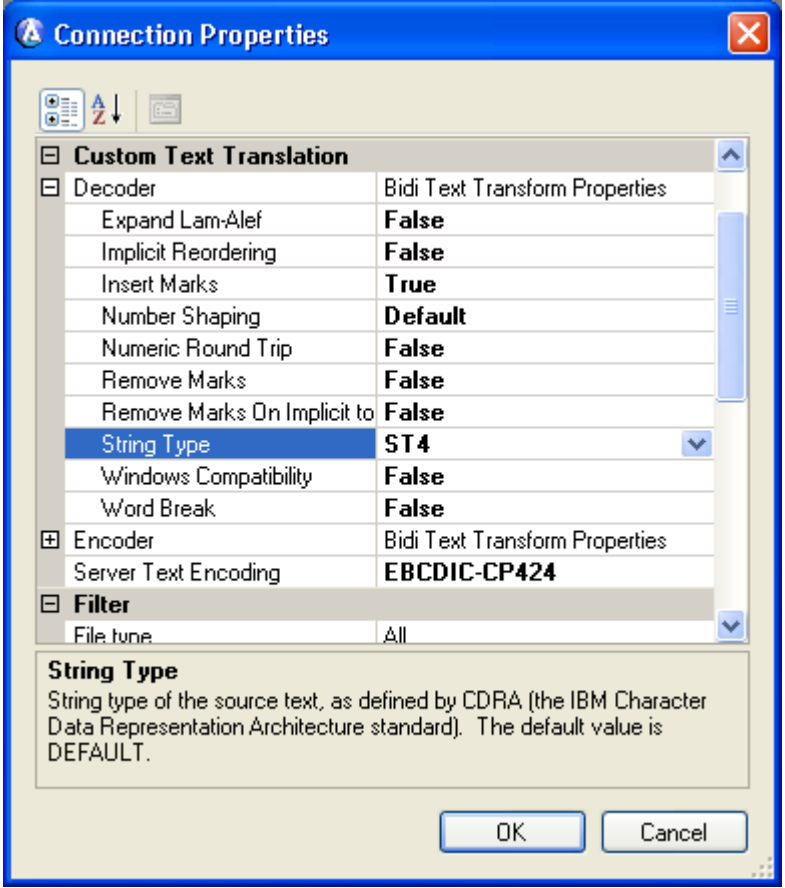
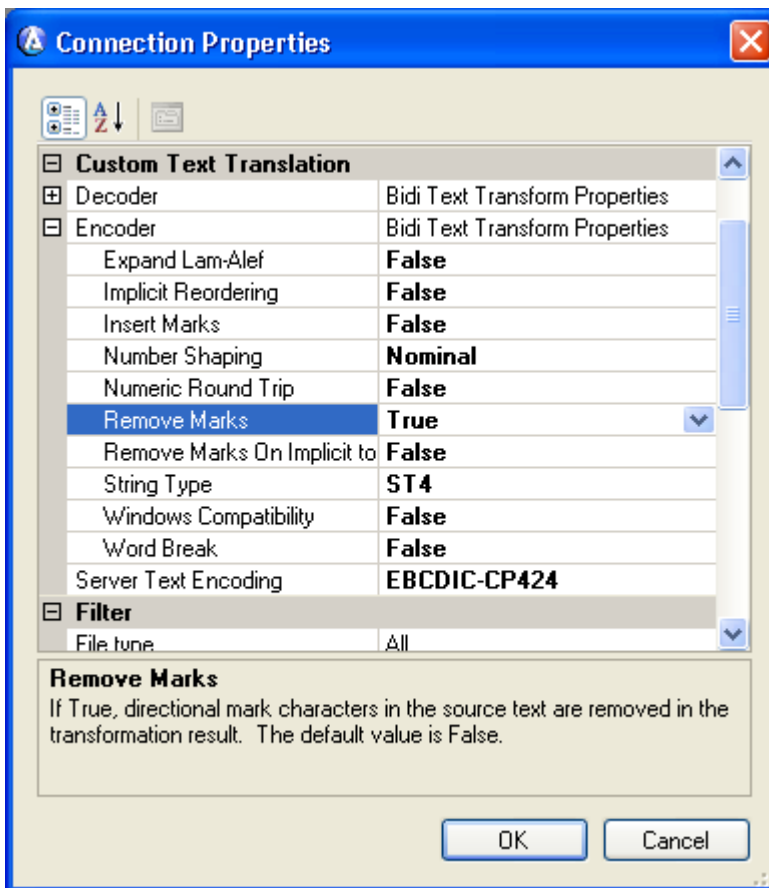


Figure 4. This example configures an EBCDIC-to-Unicode converter (Decoder) using CCSID 424, string type 4, and the Insert Marks directive.

In Figure 4, the dialog is enlarged to show all the properties for Decoder (you may do so by clicking and dragging the “size grip” control in the lower-right corner of the dialog window).

In this example, two properties are changed: String Type and Insert Marks (for a complete explanation of the properties of the ASNA bidi provider's Decoder and Encoder, see the Appendix A). The String Type has been changed from "DEFAULT" to ST4, and the Insert Marks property is enabled.

Likewise, the Encoder must be configured. Notice in Figure 5 that the properties of Encoder are identical to those of Decoder, but the values are not. In some cases, a characteristic of a transformation result of Decoder should be accounted for by the Encoder. In this example, text transformed by Decoder's enabled Insert Marks property may include special Unicode characters called "direction marks". So the configuration in Figure 5 enables the Remove Marks property so that direction mark characters effect the transformation from Unicode to EBCDIC, but are not included in the result.



*Figure 5. Remove Marks is enabled in the Encoder, so that text obtained through the Decoder configuration in Figure 4 will have the direction marks removed when converting back to EBCDIC.*

## Appendix A – Implementing and Configuring an Alternate Server Text Translation Provider

This appendix assumes that the reader has at least a basic understanding of XML and the use of .NET configuration files. For a thorough introduction to .NET configuration files, please consult the [Configuration Files](#) topics in the .NET framework documentation.

To configure your DataGate application to use the Bidi plug-in (or any other alternate encoding engine), the .NET configuration file *must contain* an `<applicationSettings>` element referring to the factory class' assembly-qualified name. For example, the following is a .NET application configuration file containing only the elements required to configure the plug-in.

*Note that some of these elements may already be present in an existing configuration, but the `<ASNA.DataGate.Client.Properties.AltEncodingProps>` element **must be unique in the file**.*

```
<?xml version = "1.0"?>
<configuration>
  <configSections>
    <sectionGroup name="applicationSettings"
type="System.Configuration.ApplicationSettingsGroup, System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" >
      <section
name="ASNA.DataGate.Client.Properties.AltEncodingProps"
type="System.Configuration.ClientSettingsSection, System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
requirePermission="false" />
    </sectionGroup>
  </configSections>
  <applicationSettings>
    <ASNA.DataGate.Client.Properties.AltEncodingProps>
      <setting name="EncodingFactoryImpl" serializeAs="String">
        <value>
          ASNA.DataGate.DataLink.Bidi.EncodingFactory,
          ASNA.DataGate.DataLink.Bidi, Version=9.1.0.0, Culture=neutral,
          PublicKeyToken=f1b0bf4120ba8afa
        </value>
      </setting>
    </ASNA.DataGate.Client.Properties.AltEncodingProps>
  </applicationSettings>
</configuration>
```

To apply these settings to all DataGate programs that run on the computer, you can merge the above settings into the “**machine.config**” file. Machine.config can be found in the **%runtime install path%\Config** directory, where %runtime install path% is the directory containing the version of the .NET framework in use. For the .NET 2.0 framework, this directory is typically “**C:\Windows\Microsoft.NET\Framework\v2.0.50727\Config**”.

*Before editing machine.config, make a backup copy. Use **extreme caution** when editing machine.config; as the settings here effect **all .NET applications** on the computer.*

Note that machine.config already contains a `<configSections>` element, but that you will probably need to add the `<sectionGroup>` element above (and its child elements), as a child of `<configSections>`. If it already exists in machine.config, add the above `<section>` element to it as a child.

Likewise, add the `<applicationSettings>` element above (and its children) to the `<configuration>` element of machine config, unless it already exists as a **System.Configuration.ApplicationSettingsGroup** type. If `<applicationSettings>` already exists, add the `<ASNA.DataGate.Client.Properties.AltEncodingProps>` element above (and its children) to it.

If another `<applicationSettings>` element already exists in machine.config, but does not have `'type="System.Configuration.ApplicationSettingsGroup...'`, then give the above `<applicationSettings>` element a different name; such as `<applicationSettingsForASNA>`.

The following listing is an example of a name collision with a pre-existing `<applicationSettings>` element that is resolved by renaming the ASNA element. Omitted portions of machine.config are denoted below by the “...” ellipsis:

```
<?xml version="1.0"?>
<configuration>
  <configSections>
    ...
    <sectionGroup name="applicationSettings"
type="SomeOtherApp.SettingsGroup, SomeOtherApp, Version=1.0.0.0" >
    ...
  </sectionGroup>
  ...
  <sectionGroup name="applicationSettingsForASNA"
type="System.Configuration.ApplicationSettingsGroup, System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" >
    <section
name="ASNA.DataGate.Client.Properties.AltEncodingProps"
type="System.Configuration.ClientSettingsSection, System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
requirePermission="false" />
  </sectionGroup>
</configSections>
  ...
  <applicationSettingsForASNA>
    <ASNA.DataGate.Client.Properties.AltEncodingProps>
      <setting name="EncodingFactoryImpl" serializeAs="String">
        <value>
          ASNA.DataGate.DataLink.Bidi.EncodingFactory,
          ASNA.DataGate.DataLink.Bidi, Version=9.1.0.0, Culture=neutral,
          PublicKeyToken=f1b0bf4120ba8afa
        </value>
      </setting>
    </ASNA.DataGate.Client.Properties.AltEncodingProps>
  </applicationSettingsForASNA>
</configuration>
```

```
        </setting>
      </ASNA.DataGate.Client.Properties.AltEncodingProps>
    </applicationSettingsForASNA>
    ...
  </configuration>
```

To specify that a different encoding engine be used, say for a new version of **ASNA.DataGate.DataLink.Bidi**, the only change required to the above configuration is the text content of the `<value>` element.