Considerations for Migration for ASNA Monarch® 6.0

Contents

Considerations for Migration Guide

Monarch Overview

ASNA Monarch® is a set of technologies that assist in the migration of RPG applications to the .NET platform. While it strives to provide seamless flow of the applications objects to the new platform, there are several considerations that must be undertaken and instances where complete like-to-like transformation is not possible.

This document provides information on the level of transparency and support achieved by Monarch 6.0 for the different object types. Strategies are suggested for dealing with those issues where manual intervention is required.

Monarch will continue to evolve as market requirements change; therefore, the information provided in this document is subject to change. This information is current as of June 2013.

One of the recent additions to Monarch runtime is the ability to call interactive programs on the IBM i and have the 5250 output displayed in the browser where the migrated Monarch display files are shown.

RPG Programs

Op-Codes

As of version 6.0, all RPG op codes are supported with the exception of those shown in the table below.

Unsupported	Unsupported
ACQ	NEXT
ALLOC	POST
DEALLOC	REALLOC
DEBUG	REL
DSPLY	RESET
DUMP	SHTDN

Features

Overlapping Fields

The overlapping data structure field of RPG is used to remap areas of memory that allow the program to give multiple meanings to the same bytes. .NET strictly enforces type safety and abhors this kind of arbitrary reinterpretation of memory. Remapping is sometimes used in a very conflicting manner but there are certain common usages that are convenient and valid.

We have identified several common idioms involving overlapping data fields, for instance giving individual field names to elements of an array, and masquerading a data structure as a large character field - typically stored as an unformatted record or as a data area. AVR provides the commands *DclAliasGroup* and *DSLoad*/*DSDump* to facilitate these types of operations.

There is another common usage of remapping, extensively used across the RPG community, which requires special handling. It involves partitioning a text field into subfields. The most prevalent and probably perverse use of this style involves dealing with dates. The Date (and Time) type did not appear in RPG until the mid nineties with the advent of RPG ILE. Prior to that, RPG programmers were forced to use a Character (or Decimal) type to hold a date field, and then subdivide the field into the year, month, and day subfields. Another case of subfields deals with encoded data, for example composing an account number based on the division, department, and project of the account. Monarch makes use of the AVR command **DclAlias** and the **Overlay** keyword of **DclDsFld** to implement overlapping fields.

Overlapping data structure fields that are not wholly contained in a parent field are flagged as errors that have to be dealt with by hand. The technique to be used in these cases is the establishment of a property where any necessary concatenation and sub-stringing is done to produce the desired subfield.

Character vs. Byte

In AVR, character fields are kept in Unicode (as is standard in .NET languages), so a single character occupies two bytes.

In RPG, a single character occupies a single byte, and some programs take advantage of this fact to the point that some of their algorithms blur the difference between a byte and a char; these algorithms many times also depend on the EBCDIC encoding of character fields and assume to know the bit patterns used in their representation.

Programs have these dependencies need to be revised once converted to AVR so that their EBCDIC and single byte dependencies are eliminated.

Hexadecimal Constants

Constants of the form X'xxxx' are invalid, however, the H'xxxx' are valid.

Cycle

Monarch supports the RPG Cycle. The features include: Primary and secondary files, control level breaks and matching records. However, halt indicators are not supported.

Internally Described Files

Monarch provides support for internally described print files (please see print files below), however there is no support for internally described workstation and disk files.

ILE RPG

Monarch supports single and multi-module ILE RPG programs and Service programs. Procedures are also supported.

Menus

Monarch supports only UIM menus as entry points for GamePlans.

Program Status DS and File Information DS

A program-status data structure provides program exception (error) information to the program. The PSDS is defined in the main source section; therefore, there is only one PSDS per module.

The location of the subfields in the PSDS is defined by special keywords or by predefined 'From' and 'To' positions. In order to access the subfields, you assign a name to each subfield. The following fields are supported on the Program Status DS:

Offset/Keyword	Description
1	Program
191	Job Start Date
244	Job
254	User
264	Job Number
270	Job Start Time
276	Program Start Date
282	Program Start Time
358	Current User
*PROC	Program
*PROGRAM	Program

The following fields are supported on the File Information DS:

Offset/Keyword	Description
1	File Name
38	Record Name
83	For disk file, 'actual' File Name used
93	For disk file, File's Library
156	For disk file, Record Count
261	Record
367	Flags
369	For workstation file, AID Byte For print file, Page Count
370	For workstation file, Cursor Row
371	For workstation file, Cursor Column
376	For workstation file, Subfile Relative Record Number
378	For workstation file, Subfile Lowest Relative Record Number

397	For disk file, Relative Record Number
*FILE	File
*RECORD	Record Name

CL Programs

A CL Program has the capability of using over a thousand commands provided by OS/400, plus any number of user-created commands. Usage of CL can be divided into two major categories: **System Administration** and **Application Coordination**.

The administration of the **system** involves operations like the creation of user profiles and the save/restore procedures. These activities are not considered part of the application itself and Monarch does not attempt to facilitate such activities. Normal Windows techniques must be employed to affect these kinds of activities.

On its other personality, CL is used to set up the environment for RPG programs to run. Typical functions done by CL in this context are:

- Set up the Library List
- Override database file
- Select a different File or Member to be used by the RPG 'F' spec
- Establish a Query File
- Maintain application parameters in data area
- Allocate objects
- Control the Program Message Queue

These actions affect the OS/400 Job where programs are running and have an effect on all programs on the same job. Monarch Framework provides these facilities through a set of classes that supplement the .NET Framework.

Commands Supported

The following list of commands is supported[1] by Monarch 6.0.

ADDLIBLE	ENDPGM
ADDPFM	GOTO
ALCOBJ	ENDCMTCTL
CALL/CALLB	IF
CALLPRC	INZPFM
CHGDTAARA	MONMSG
CHGJOB	OPNQRYF
CHGVAR	OVRDBF
CHKOBJ for ObjType *FILE only	OVRDSPF
CLOF	OVRPRTF
CLRPFM	PGM
COMMIT	QCLSCAN
CRTDTAARA	RCVF
CRTDUPOBJ for files only	RETURN
DCL	RMVLIBLE
DCLF	RMVMSG
DLTF FILE (library/file name) only	ROLLBACK
DLCOBJ	RTVDTAARA
DLTDTAARA	RTVJOBA
DLTF	SBMJOB

[1] Not all keywords are supported on all of the commands listed.

DLTOVR	SNDF
DO / ENDDO	SNDPGMMSG
ELSE	SNDRCVF
	STRCMTCTL

Display Files

Display files are migrated into Active Server Pages for .NET forms. The migration strategy for each display file is to create an ASP.NET form composed of two files. First, there is an .aspx file containing HTML, ASP, and Monarch elements describing the layout and data composition of the display using server controls provided by Monarch.

Second, an .aspx.vr file contains the code-behind class, which extends the ASNA.Monarch.WebDspF.Page class. This class is part of a framework providing the runtime support to make the display file appear to the user.

The Display File Agent creates the ASP.NET form taking as input the OS/400 display files DDS specifications. Most of the elements found in the DDS specification are migrated. However, there are two general features, which today are not dealt with: Help and window widgets. There are multiple keywords that form these two groups. The help system defined in the DDS is alien to the web model and it is not migrated. In the mid 90s, DDS incorporated a set of keywords to facilitate the creation of Windows-looking screens. Because specialized hardware was needed to render these keywords properly, the practice of using them never caught on. The exception is the support for the WINDOW keyword that is converted to a browser pop-up window.

Monarch provides modern user interface elements such as calendar control, drop-down lists, button images that can be easily used to replace window widget DDS types.

CssClass Property Migration Considerations

During migration, the Solution Builder in Monarch Cocoon updates the CssClass Property. Some common values[2] are as follows:

- **DdsKey** - used on the function key buttons.
- **MessageLine** - used on the message line used for displaying input errors.
- **DdsErrorReset** - used on the button shown when there are input errors.
- **DdsRecord** - used as a Div wrapping record.
- **DdsSflMsgField** - a character field used for Message Subfiles.
- **DdsCharField** - used for all other character fields.
- **DdsDecField** - used for all decimal fields.
- **DdsConstant** - used for literal fields.
- **DdsDateField** - used for date fields.
- **DdsDecDateField** – used for decimal date fields.
- **DdsTimeField** – used for time fields.
- **DdsTimestampField** – used for timestamp fields.

If a field is found to be in error, (badly typed input or user turned indicator on marking it as in error), the value of the CssClass property is suffixed with **_Error**. In addition, a right justified field will have **_Right** appended, while those that are output only have **_OutputOnly** appended.

Since users can modify the CssClass property, assume a character field has been modified to have a class name of MyClass, then when found in error, it would be generated to the browser with a style of **class=MyClass_Error**.

[2] Always refer to the latest Cocoon .css files for the current styles.

Refer to **Display File Template and Cascading Style Sheet Considerations** in the Cocoon User's Guide for more information on the .css files provided and also how to override these styles to utilize any existing style sheets you already have established in your applications.

Appendix I shows individual display file keywords and their level of support.

Printer Files

Two techniques are employed by RPG report applications to describe the layout of the report; **externally-described printer files** and the use of **O-Specs** to internally describe the layout. Monarch provides two migration agents to deal with these two techniques. Both agents target the DataGate Printer File facilities. These facilities consist of:

- Print Files
- Print File Designer
- Data-aware controls
- Render engine

The **printer O-Spec agent** takes the RPG O-Specs as input and generates a DataGate Print File. Print File Overflow Areas are also supported. All the fields associated with an exception record are grouped together into an externally described record format. If multiple exception records exist with the same name, each one generates its own record format and the name of each is distinguished by the indicators that control it.

The **print file agent** takes DDS specifications as input and creates a DataGate Print File. **Appendix II** specifies the level of support provided by Monarch for Printer file DDS keywords.

When each record format is printed, the print position within the page always advances the height of the format; there is no mechanism available to print two record formats on the same area of the page.

While DataGate print fields can be printed (or not) based on conditional indicators, fields and labels do not have the ability to be underlined or bolded via conditional indicators, instead the underline or bold properties must be set programmatically in AVR.

Database Files

Client/Server *vs*. Host Based

Traditional, green-on-black iSeries applications have the advantage of being run on the same machine as the database engine and data store. The foremost constraint in client/server processing is the network. Software engineers from all disciplines know that with client/server they are dealing with the physical aspects of packet transfer, task switching, band-width, traffic congestion, and so on. Client/server is well suited to transaction processing, especially when these features can be employed.

Batch and batch-like processing may be problematic with client/server because of the high data volumes and the high number of I/O requests. When the performance of batch processing becomes an issue considerably affecting the application's user experience - the Migrator needs to manually improve the program's performance.

Improving Batch Processing Performance

Depending on the algorithm used by the legacy host-based batch processing program, one or more of the following techniques may need to be manually applied; use net-blocking on input only files, reduce the number of times loops execute, and reduce – or eliminate – file operations that use keys (like CHAIN) by utilizing join files.

Moving to SQL Server

If data files are to be moved to SQL Server to be used at runtime via DataGate for SQL Server (DSS), then DSS restrictions apply to the migrated application.

Even though DSS tries to make *SQL Server* look like DB2/400, there are several features that can not be implemented in a totally transparent fashion.

The following items are probably the ones with the most impact on the migrated code.

Dealing with Applications using Multi-member Files when moving to SQL Server

DataGate for SQL Server (DSS) does not implement the concept of a multi-member file. When migrating an application that makes use of a multi-member file, it is necessary to modify the application to use multiple files instead of multiple members. Instead of using the add member methods, one must use the copy file methods.

Under DSS, each file is implemented by a table or view depending upon whether the file is physical or logical. On the iSeries, a physical file has several attributes such as the record format definition, which is an ordered collection of fields, an optional key set at zero or more member that are the actual containers of data records. By contrast, a SQL Server table also defines a collection of columns and contains a set of rows, but the concept of data members is nonexistent.

For illustration purposes, let us assume that the application uses a file called **Sales** with one member for each month of the year labeled January through December. Furthermore, at the beginning of each month, the application adds a new member to the file where the sales for the month are accumulated.

One approach is to create a file with no data that will serve as the model for all the files that will contain the actual data taking the place of the multiple members. Let's call the model file **Sales__Model**. For each member, a copy of the model file is created using the member name as a suffix to make the name unique.

The application uses the appropriate suffixed file to store data, relegating the model file only as a template for the record format and key description. In our example, at the start of a new month, say September, we would copy Sales__Model to Sales_September. Notice that by using a double underscore for the Model file and a single underscore to separate the file name from the suffix (member name) all related files list together under Database Managers.

If there are logical members associated with the application, a similar approach must be followed taking into account that the base file will have to be modified to accommodate the corresponding new combined file_member name. Also, notice that the longer concatenated name will most likely exceed the 10-character limit imposed by OS/400, but allowed under DataGate for SQL Server.

It is important to use DataGate facilities to create the suffixed files because a physical file is more than just a plain table and a logical file is more than just a view. Special consideration has to be given to the key that ends up being an index on the table, and there are extended properties that have to be added to the table/view to preserve other attributes like column headings and text.

An alternative implementation is the usage of partitioned table. Under this scheme, a column (say MemberName) is added to the file where the 'name' of the member is to be stored for each of the records in the file. Use this MemberName in the partition function of the table and views to separate the data amongst the different 'members' of the file. Kimberly L. Tripp has written a white paper introducing the concepts of partitioned tables; you can find it at the address that follows.

http://www.sqlskills.com/resources/Whitepapers/Partitioning_in_SQL_Server_2005_Beta_II.htm

Lack of support for Multi-member Files under SQL Server

DSS for .NET implements a physical file through the use of a native SQL Server table. A logical file is implemented through a native view. SQL Server Views are single formatted in nature, so there is no support for multi-format logical data files. You will have to eliminate any reference to multi-format logical files in your application

> ***Note:*** *Print files, which are typically multi-format, are fully supported in DSS.*

Logical Field Restrictions when moving to SQL Server

There are two restrictions on the usage of logical fields when they change the name or the type of their corresponding base physical field. When the field is retyped, most typically because the field is a concatenation or substring of the physical, then the field becomes read-only. A logical field whose name has changed from its physical base field cannot be used as a key field in the logical file.

Unlocking Records when moving to SQL Server

This next issue is probably the most demanding area of application adaptation. The problem arises in two areas: using the Unlock keyword on the read operations and on the implementation of the operation Unlock.

DSS uses *SQL Server* Cursors to implement file access. When a file is opened for update, it is not possible to tell *SQL Server* to not lock the record on a read, so this option is not valid for files opened for Update. You have two options to solve this problem: Declare a second instance of the file marked as input only and use it wherever the NoLock option was given on a read/chain. The other alternative is to leave the NoLock and follow the **READ/CHAIN** with an unlock (see the unlock challenge in the following paragraph).

The other problem is in the use of the Unlock operation. Unlock leaves the cursor in a no-position state, meaning you can not perform a subsequent read (next/previous) without repositioning the file with a SET (SETLL/SETGT) or CHAIN.

Advanced Techniques to Optimize Batch Processing Performance when moving to SQL Server

In addition to the techniques described above that apply to any Client/Server database migration, when targeting Microsoft SQL Server, other advanced techniques may be applied to further improve performance; such as manually converting the legacy ISAM (Index Sequential Access Method) to Recordset based access using ADO and using SQL stored procedures.

Appendix I - DDS Display File Keywords

The following tables specify the level of support provided by Monarch for Display file DDS keywords.

Display Field-Level Keywords

Supported	Supported in future release	Not applicable	Unsupported
AID	CHGINPDFT	AUTO	CHRID
ALIAS	CHKMSGID	BLKFOLD	DUP
BLANKS	DLTCHK	CMP	EDTMSK
CHANGE	DLTEDT	OVRATR	ENTFLDATR
CHECK	DFTVAL	OVRDTA	NOCCSID
COLOR	FLDCSRPRG	PUTRETAIN	
COMP	FLTFIXDEC	WRDWRAP	
CNTFLD	FLTPCN		
DATE	HTML		
DATFMT	INDTXT		
DATSEP	MAPVAL		
DFT	VALNUM		
DSPATR[3]			
EDTCDE			
EDTWRD			
ERRMSG			
ERRMSGID			
LOWER			
MSGCON			
MSGID			
RANGE			
REFFLD			
SYSNAME			
TEXT			
TIME			
TIMFMT			
TIMSEP			
USER			
VALUES			

[3] See table of display attributes supported later in this section.

Display Record-Level Keywords

Supported	Supported in future release	Not applicable	Unsupported
ALTNAME	CHGINPDFT	OVRATR	ALARM
CAnn	CLRL	OVRDTA	ALWGPH
CFnn	HELP	PUTOVR	ALWROL
CHANGE	INDTXT	PUTRETAIN	ASSUME
CHECK	LOGINP	RMVWDW	BLINK
CSRLOC	LOGOUT	WDWBORDER	CCSID
ERASE	VALNUM	WRDWRAP	CLEAR
OVERLAY			CSRINPUT
PAGEDOWN			CSRINPONLY
PAGEUP			DSPMOD
PRINT			ENTFLDATR
PROTECT			ERASEINP
RTNCSRLOC [4]			FRCDTA
RTNDTA			GETRETAIN
ROLLDOWN			HOME
ROLLUP			INVITE
SETOF(F)			INZINP
SLNO [5]			INZRCD
TEXT			KEEP
VLDCMDKEY			LOCK
WDWTITLE			MDTOFF
WINDOW			MSGALARM
			RETCMDKEY
			RETKEY
			RETLCKSTS
			UNLOCK
			USRDFN

[4] Support is not provided for cursor positioning **within** a field.
[5] Support is not provided for (*VAR)

Display File-Level Keywords

Supported	Supported in future release	Not applicable	Unsupported
CAnn	ALTHELP	INDARA	ALTPAGEDWN
CFnn	CHGINPDFT	WRDWRAP	ALTPAGEUP
CHECK	DSPRL		ALWGPH
ERRSFL	HELP		CLEAR
PAGEDOWN	INDTXT		CSRINPONLY
PAGEUP	MSGLOC		DSPSIZ
PRINT	VALNUM		ENTFLDATR
REF			HOME
ROLLDOWN			INVITE
ROLLUP			MSGALARM
VLDCMDKEY			OPENPRT
WDWBORDER			PASSRCD
			PRINT(Lib)/File)
			USRDSPMGT

Subfile Keywords

Supported	Supported in future release	Unsupported
SFL	SFLINZ	SFLCSRPRG
SFLCLR	SFLRNA	SFLDLT
SFLCTL		SFLENTER
SFLCSRRRN		SFLROLVAL
SFLDROP		SFLSCROLL
SFLDSP		
SFLDSPCTL		
SFLEND		
SFLFOLD		
SFLLIN		
SFLMODE		
SFLMSG		
SFLMSGID		
SFLMSGKEY		
SFLRCDNBR		
SFLSMGRCD		
SFLNXTCHG		
SFLPAG		
SFLPGMQ		
SFLSIZ		

Display Attributes for DSPATR Keyword

Supported	Supported in future release	Unsupported
ND-Non Display	HI-High Intensity	BL-Blinking Field
PC-Position Cursor		CS-Column Separator
PR-Protect		MDT-Set Change Data Tag
		OID-Operator Identification
		SP-Select by Light Pen
		RI-Reverse Image
		UL-Underline

Appendix II - DDS Print File Keywords

The following tables specify the level of support provided by Monarch for Print file DDS keywords.

Printer Field-Level Keywords

Supported	Supported in future release	Not applicable	Unsupported
ALIAS	FLTFIXDEC	BLKFOLD	CDEFNT
BARCODE	FLTPCN		CHRID
COLOR	INDTXT		CHRSIZ
DATE	TEXT		CPI
DATFMT			CVTDTA
DATSEP			DLTEDT
DFT			DTASTMCMD
EDTCDE			FNTCHRSET
EDTWRD			PRTQLTY
FONT[6]			TRNSPY
FONTNAME			TXTRTT
HIGHLIGHT			
MSGCON			
PAGNBR			
POSITION[7]			
REFFLD			
SKIPA			
SKIPB			
SPACEA			
SPACEB			
TIME			
TIMFMT			
TIMSEP			
UNDERLINE			

[6] Font “PointSize” height value supported only.

[7] Position supported when position-down and position-across are expressed as constants.

Printer Record-Level Keywords

Supported	Supported in future release	Not applicable	Unsupported
CPI	BOX	OVERLAY	CDEFNT
ENDPAGE	DRAWER		CHRSIZ
FONT	DUPLEX		DFNCHR
FONTNAME	INDTXT		DOCIDXTAG
HIGHLIGHT	LINE		DTASTMCMD
LPI	OUTBIN		ENDPAGGRP
PAGNBR	TEXT		FNTCHRSET
SKIPA			FORCE
SKIPB			GDR
SPACEA			INVDTAMAP
SPACEB			INVMMAP
			PAGRTT
			PRTQLTY
			STRPAGGRP
			ZFOLD

Printer File-Level Keywords

Supported	Supported in future release	Not applicable	Unsupported
REF	INDTXT	INDARA	DFNCHR
			FNTCHRSET
			SKIPA
			SKIPB