

# IBM i 7.1 Overview – Development Tools



## CL Programming

- Retrieve the CL source for an Integrated Language Environment (ILE) module (CRTCLMOD or CRTBNDCL with ALWRTVSRC(\*YES)) parameter
- Can be a module \*MODULE object or a module within a ILE program \*PGM or service program \*SRVPGM
- Declare CL variable (DCL) command LEN(8) for signed and unsigned integer (\*INT and \*UINT) – only for objects compiled using the CRTCLMOD or the CRTBNDCL command
- Show DO and SELECT nesting levels in compiler listing – parameter (OPTION(\*DOSLTLVL))
- Nested INCLUDE support (option to retrieve sources from included program source)

# RPG ILE

## XML-INTO Support

- **The *datasubf* option allows you to name a subfield that will receive the text data for an XML element that also has attributes**
  - When an XML element has attributes or child elements, the XML element (emp) itself matches an RPG data structure and the attributes (type, id) match the subfields. (Child elements would also match subfields.) If there is text data for the data structure (“John Smith”), there is no XML name available for subfield matching.
  - Without the *datasubf* option, two XML-INTO operations would be needed to get the data from that XML element.
    - One XML-INTO would get the type and id values
    - One XML-INTO would get the value “John Smith”
    - The *allowextra=yes* option would be needed for both XML-INTOs
- **The *countprefix* option reduces the need for you to specify the *allowmissing=yes* option. It specifies the prefix for the names of the additional subfields that receive the number of RPG array elements or non-array subfields set by the XML-INTO operation**

## %LEN(varying : \*MAX)

- **Get the maximum number of characters for a varying field.**

```
D char_varying      S          5000A  VARYING
  /free
  // Pass address and length of the data
  conv (%addr(char_varying : *data)
        : %len(char_varying : *max));
```

- **Problem solved by %LEN(\*MAX):**

- Starting in V6R1, calculations involving the size of the varying-length prefix became error-prone because the size might be either 2 or 4
- In V6R1, %ADDR was enhanced with %ADDR(\*DATA) to simplify getting the address of the data part. Previously, programmers added 2 to %ADDR
- %LEN(\*MAX) simplifies getting the maximum length of the data part. Previously, programmers subtracted 2 from %SIZE, and then divided by 2 for UCS-2 and DBCS

## %SCANRPL

- **The %SCANRPL built-in function replaces all occurrences a string with another string**

```
fileErr = 'File &1 not found. Please create &1.';  
msg = %scanrpl ('&1' : filename : fileErr);
```

```
// msg = 'File MYFILE not found. Please create MYFILE.'
```

- **Problem solved by %SCANRPL:**
  - Hand-written versions of scan-and-replace tend to be error prone and difficult to maintain

## %PARMNUM

- The %PARMNUM built-in function returns a parameter's position in the parameter list

```
D myProc      pi    10A  OPDESC
D company          25A  OPTIONS(*VARSIZE)
D city            25A  OPTIONS(*VARSIZE)
```

- Problem solved by %PARMNUM:

- Parameter-information APIs such as CEEDOD or CEETSTA require a parameter's number
- Soft-coding the parameter's number makes the code easier to read and maintain

```
CEEDOD (2 : more parms);           // hard to understand
CEEDOD (%PARMNUM(city) : more parms); // better
```

## Implicit CCSID conversion - parms

- **Implicit CCSID conversion is now supported for prototyped parameters passed by VALUE and by read-only reference (CONST)**
- **Reduces the code changes that have to be made when a database field is changed from alphanumeric or DBCS to Unicode (UCS-2 or UTF-16)**
- **Parameter conversion also provides a way to have a single string-handling procedure that can handle alphanumeric, UCS-2 or DBCS.**
- **In the example below, there is only a “makeTitle” procedure with a UCS-2 parameter and return value. If the passed parameter is alpha or DBCS, it will be converted to UCS-2 on the call. The procedure will work with the UCS-2 parameter and return a UCS-2 value. This returned value can then be converted on assignment to alpha or DBCS, if necessary**

```
// makeTitle() upper-cases the parameter
// and centers it within the provided length
alphaTitle = makeTitle(alphaValue : 50);
ucs2Title = makeTitle(ucs2Value : 50);
dbcsTitle = makeTitle(dbcsValue : 50);
```

## Sort and search data structure arrays

- **Sort a data structure array using one subfield as a key**

```
// sort by name
SORTA info(*).name;
// sort by due date
SORTA info(*).dueDate;
```

- **Search a data structure array using one subfield as a key**

```
// search for a name
pos = %LOOKUP('Jack' : info(*).name);
// search for today's date
pos = %LOOKUP(%date() : info(*).dueDate);
```

## Sort ascending or descending

- **Non-sequenced arrays can be sorted either ascending or descending**

```
D meetings      S      D DIM(100)
```

```
/free
```

```
// sort descending, with the most recent date first
```

```
sorta(d) meetings;
```

## Performance returning large values

- **RTNPARM keyword greatly improves performance when a procedure returns a large value**
- **The speed of using a parameter with the convenience of using a return value**
- **Especially noticeable when the prototyped return value is a large varying length value**

```

D center          pr          100000a   varying
D                                     rtnparm
D  text          50000a   const varying
D  len           10i 0   value
D title          s          100a   varying
/free
  title = center ('Chapter 1' : 60);

```

- **With RTNPARM the return value is handled as a parameter, under the covers**
- **The hidden parameter is the first parameter, so be sure to use %PARMNUM(parmname) when checking for unpassed parameters**

## Optional prototypes

- **If a program or procedure is not called by another RPG module, it is optional to specify the prototype.**
- **Here are some programs and procedures that do not require a prototype**
  - An exit program, or the command-processing program for a command
  - A program or procedure that is never intended to be called from RPG
  - A procedure that is not exported from the module

```
H main(hello)
P hello          b
D                pi          extpgm('HELLO')
D  name          10a         const
  /free
    sayHello();
...
P sayHello      b
  /free
    dsply ('Hello ' + name);
```

## Support for ALIAS names

- **Fields in externally described files can have a standard name up to 10 characters and an ALIAS name up to 30 characters.**
- **RPG III only allowed 6 characters, so many customers have files with cryptic names like CUSNAM, CUSADR. The files often have ALIAS names such as CUSTOMER\_NAME and CUSTOMER\_ADDRESS, that can be used in SQL queries.**
- **When ALIAS is specified, RPG will use the ALIAS name instead of the 10-character standard name.**
- **Supported on F specs for any file that will not have Input or Output specs generated. Used for LIKEREK data structures.**
- **Supported on D specs for any externally-described data structure**

# COBOL

- Support for **COMPUTATIONAL-5** or **COMP-5** (native binary data type)

COBOL Data Type	SQL Data Type	SQL Description
01 name PIC S9(4) COMP-5.	SMALLINT	16-bit signed integer
01 name PIC S9(9) COMP-5.	INTEGER	32-bit signed integer
01 name PIC S9(18) COMP-5.	BIGINT	64-bit signed integer

- Specify a non-numeric literal on the **VALUE** clause for a national data item
- **XML GENERATE** performance improvements
- Larger program support is enabled by the **CRTBND CBL / CRT CBL MOD OPTIMIZE** parameter, which now supports a new **\*NEVER** option value

## C Macro's

- **`__C99_CPLUSCMT`** indicates support for C++ style comments. You can define it when the `LANGLVL(*EXTENDED)` compiler option is in effect.
- **`__IBMC__`** indicates the version of the C compiler. It returns an integer of the form `VRM` where `V` represents the version, `R` the release and `M` the modification level. For example, using the IBM i 7.1 compiler with the `TGTRLS(*CURRENT)` compiler option, it returns the integer value 710.
- **`__ILEC400__`** indicates that the ILE C compiler is being used.
- **`__ILEC400_TGTVRM__`** is functionally equivalent to the **`__OS400_TGTVRM__`** macro.
- **`__SIZE_TYPE__`** indicates the underlying type of `size_t` on the current platform. For IBM i, it is unsigned int

## C++ Macro's

- **\_\_BOOL\_\_** indicates that the `bool` keyword is accepted.
- **\_\_cplusplus98\_\_interface\_\_** can be defined when the `LANGLVL(*ANSI)` compiler option is specified.
- **\_\_C99\_COMPOUND\_LITERAL** indicates support for compound literals and can be defined when the `LANGLVL(*EXTENDED)` compiler option is in effect.
- **\_\_C99\_FUNC\_\_** indicates support for the `__func__` predefined identifier and can be defined when the `LANGLVL(*EXTENDED)` compiler option is in effect.
- **\_\_C99\_HEX\_FLOAT\_CONST** indicates support for hexadecimal floating constants and can be defined when the `LANGLVL(*EXTENDED)` compiler option is in effect.
- **\_\_C99\_PRAGMA\_OPERATOR** indicates support for the `_Pragma` operator and can be defined when the `LANGLVL(*EXTENDED)` compiler option is in effect.
- **\_\_C99\_RESTRICT** indicates support for the C99 restrict qualifier and can be defined when the `LANGLVL(*EXTENDED)` compiler option is in effect.
- **\_\_C99\_VARIABLE\_LENGTH\_ARRAY** indicates support for variable length arrays and can be defined when the `LANGLVL(*EXTENDED)` compiler option is in effect.

## C++ Macro's

- **\_\_IBMCPP\_\_** indicates the version of the AIX XL C++ compiler upon which the ILE C++ compiler is based. It returns an integer representing the compiler version. For example, using the IBM i 7.1 compiler with the TGTRLS(\*CURRENT) compiler option, **\_\_IBMCPP\_\_** returns the integer value 900. 900 means the ILE C++ compiler is based on the XL C++ V9.0 compiler.
- **\_\_IBM\_\_ALIGN** indicates support for the **\_\_align** specifier.
- **\_\_IBM\_\_ATTRIBUTES** indicates support for type, variable, and function attributes and can be defined when the LANGLVL(\*EXTENDED) compiler option is in effect.
- **\_\_IBM\_\_COMPUTED\_GOTO** indicates support for computed goto statements and can be defined when the LANGLVL(\*EXTENDED) compiler option is in effect.
- **\_\_IBM\_\_EXTENSION\_KEYWORD** indicates support for the **\_\_extension\_\_** keyword and can be defined when the LANGLVL(\*EXTENDED) compiler option is in effect

## C++ Macro's

- **\_\_IBM\_LABEL\_VALUE** indicates support for labels as values and can be defined when the **LANGLVL(\*EXTENDED)** compiler option is in effect.
- **\_\_IBM\_LOCAL\_LABEL** indicates support for local labels and can be defined when the **LANGLVL(\*EXTENDED)** compiler option is in effect.
- **\_\_IBM\_MACRO\_WITH\_VA\_ARGS** indicates support for variadic macro extensions and can be defined when the **LANGLVL(\*EXTENDED)** compiler option is in effect.
- **\_\_NO\_RTTI\_\_** can be defined when the **OPTION(\*NORTTI)** compiler option is in effect.
- **\_\_OPTIMIZE\_\_** indicates the level of optimization in effect. The macro is undefined for **OPTIMIZE(10)**. For other **OPTIMIZE** settings, the macro is defined with 2 for **OPTIMIZE(20)**, 3 for **OPTIMIZE(30)** and 4 for **OPTIMIZE(40)**.
- **\_\_RTTI\_DYNAMIC\_CAST\_\_** can be defined when the **OPTION(\*RTTIALL)** or **OPTION(\*RTTICAST)** compiler option is specified.
- **\_\_RTTI\_TYPE\_INFO\_\_** can be defined when the **OPTION(\*RTTIALL)** or **OPTION(\*RTTITYPE)** compiler option is specified.

## C and C++ Macro's

- **\_\_BASE\_FILE\_\_** indicates the fully qualified name of the primary source file.
- **\_\_IBM\_DFP\_\_** indicates support for decimal floating-point types and can be defined when the **LANGLVL(\*EXTENDED)** compiler option is in effect.
- **\_\_IBM\_INCLUDE\_NEXT** indicates support for the **#include\_next** preprocessing directive.
- **\_\_IBM\_TYPEOF\_\_** indicates support for the **\_\_typeof\_\_** or **typeof** keyword. This macro is always defined for C. For C++, it is defined when the **LANGLVL(\*EXTENDED)** compiler option is in effect.
- **\_\_IFS\_IO\_\_** can be defined when the **SYSIFCOPT(\*IFSIO)** or **SYSIFCOPT(\*IFS64IO)** compiler option is specified

## C and C++ Macro's

- **\_\_IFS64\_IO\_\_** can be defined when the **SYSIFCOPT(\*IFS64IO)** compiler option is specified. When this macro is defined, **\_LARGE\_FILES** and **\_LARGE\_FILE\_API** are also defined in the relevant IBM-supplied header files.
- **\_\_LONGDOUBLE64** indicates that the size of a long double type is 64 bits. This macro is always defined.
- **LONG\_LONG** indicates support for IBM long long data types and can be defined when the **LANGLVL(\*EXTENDED)** compiler option is in effect.
- **\_\_POSIX\_LOCALE\_\_** can be defined when the **LOCALETYPE(\*LOCALE)** or **LOCALETYPE(\*LOCALEUCS2)** or **LOCALETYPE(\*LOCALEUTF)** compiler option is specified.
- **\_\_UCS2\_\_** can be defined when the **LOCALETYPE(\*LOCALEUCS2)** compiler option is specified.
- **\_\_UTF32\_\_** can be defined when the **LOCALETYPE(\*LOCALEUTF)** compiler option is specified.

## Pragma's

- **The *#pragma do\_not\_instantiate* directive suppresses the instantiation of a specified entity**
  - Handling template instantiations manually (that is, compiler options `TEMPLATE(*NONE)` and `TMPLREG(*NONE)` are in effect), and the specified template instantiation exists in another compilation unit, using `#pragma do_not_instantiate` ensures that you do not get multiple symbol definitions during the link step
- **The *#pragma namemanglingrule* directive provides fine-grained control over the name mangling scheme in effect for selected portions of source code, specifically with respect to the mangling of cv-qualifiers in function parameters**
  - Control whether top-level cv-qualifiers are mangled in function parameters or whether intermediate-level cv-qualifiers are to be considered when the compiler compares repeated function parameters for equivalence

## Encrypt the debug listing view (all ILE compilers)

- **The problem:**
  - You want to ship a debuggable version of your application to your customers, but you don't want them to be able to read your source code through the debug view
- **The solution:**
  - Encrypt the debug view so that the debug view is only visible if the person knows the encryption key.
    - CRTBNDRPG MYPGM DBGENCKEY('my secret code')
  - Then either
    - STRDBG MYPGM DBGENCKEY('my secret code')
  - OR
    - STRDBG MYPGM
  - and wait to be prompted for the encryption key

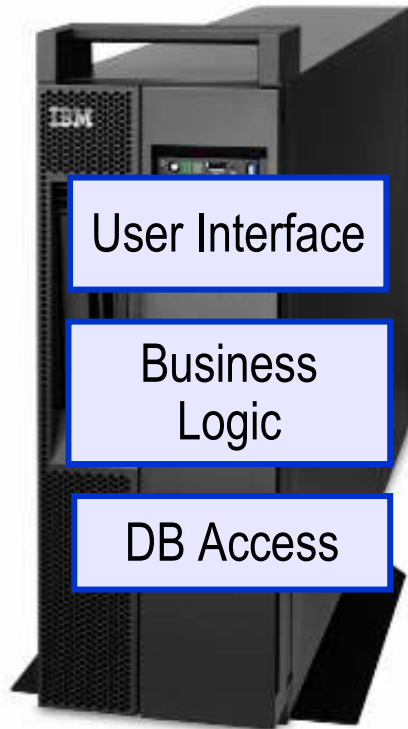
## Teraspace storage model (all ILE compilers)

- **The problems:**
  - 16MB automatic storage limits with the single-level storage model, for a single procedure, and for all the procedures on the call stack
  - RPG's %ALLOC and %REALLOC have a 16MB limit
- **The solution: use the teraspace storage model**
  - Much higher limits for automatic storage.
  - Can compile \*CALLER programs with STGMDL(\*INHERIT) so they can be called from either single-level or teraspace programs
  - RPG's %ALLOC and %REALLOC can allocate teraspace with a much higher limit
  - Teraspace allocations are the default in the teraspace storage model
  - Specify H-spec ALLOC(\*TERASPACE) to have teraspace allocations in any storage model

# Open Access for RPG

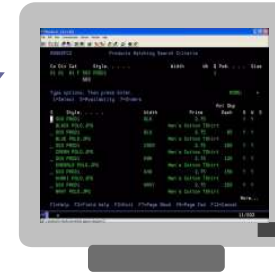
# Original Goal: Support Multiple UI

RPG / COBOL Applications



EXFMT  
EXJSP  
EXCGI  
EXEUI  
EX???

5250 Screens



JSPs

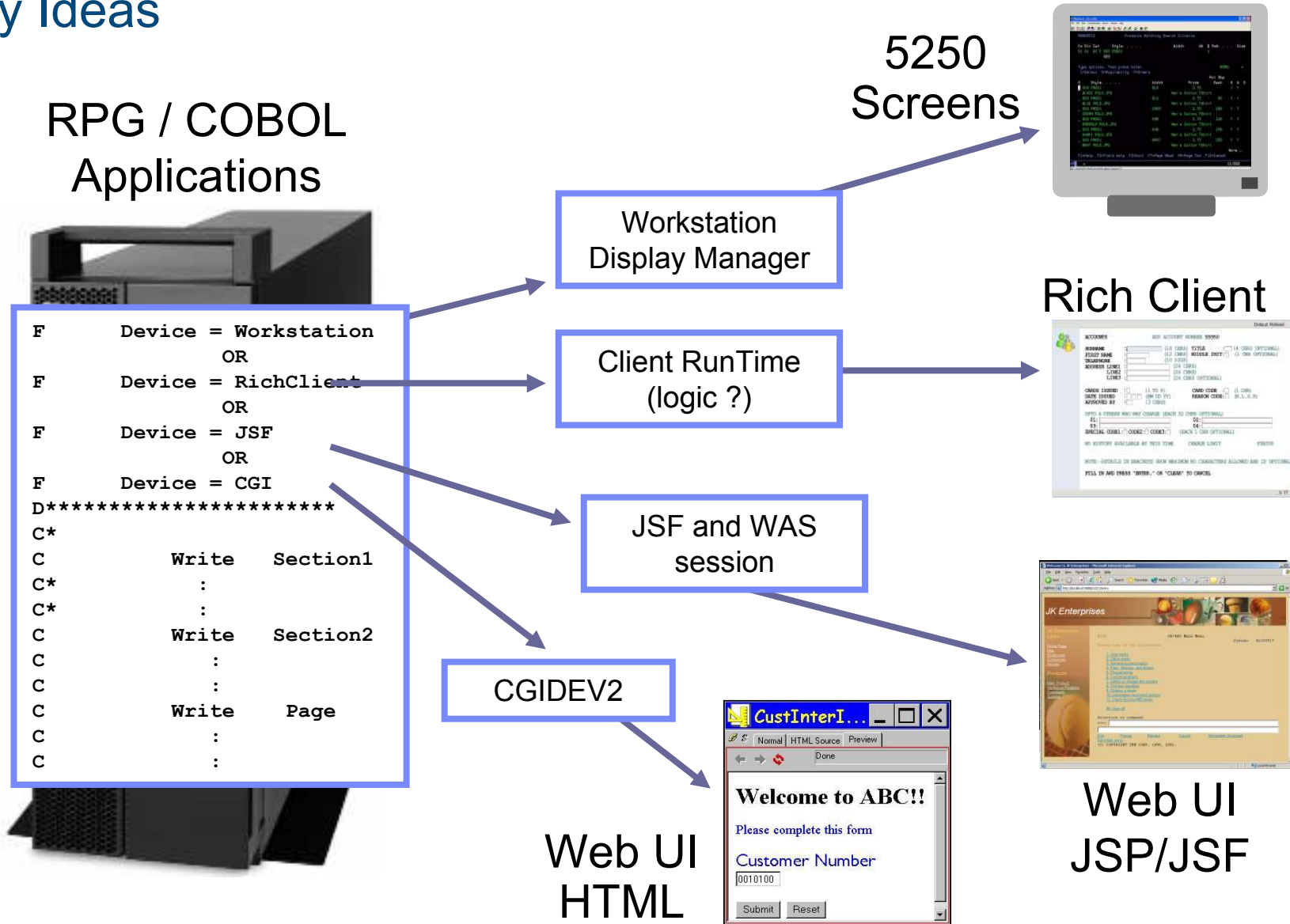


Other Servers



Mobile Devices

# Early Ideas

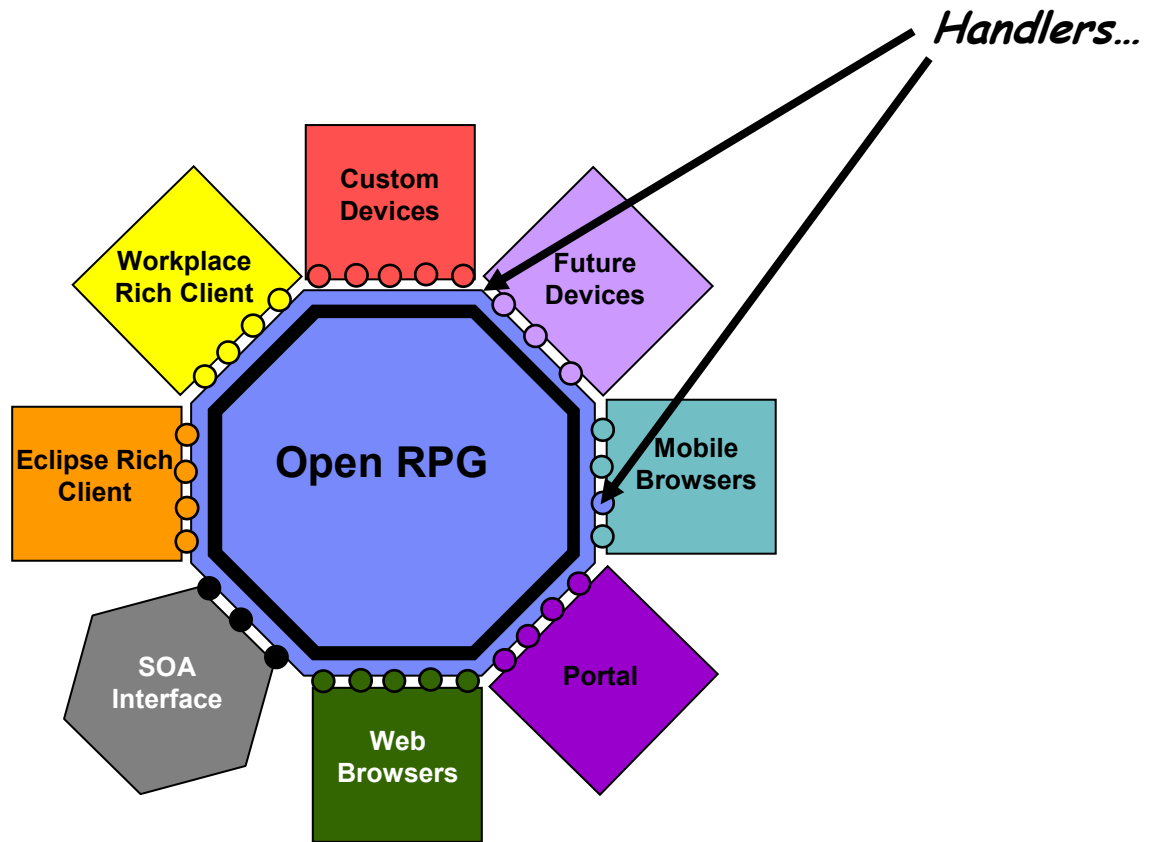






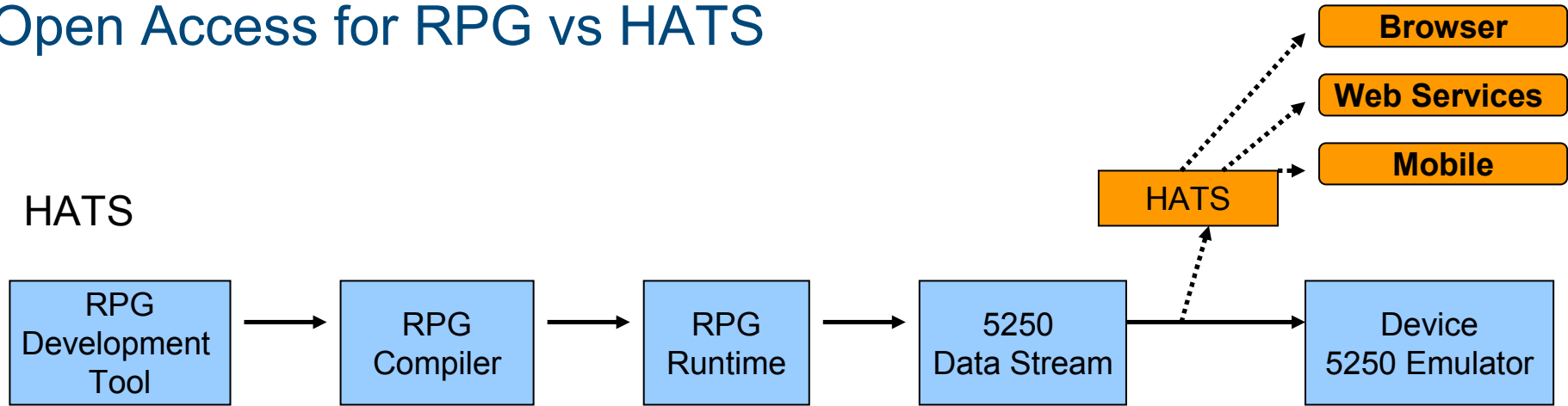
# Open Access for RPG

- Extend application reach to pervasive devices thru Open Access for RPG

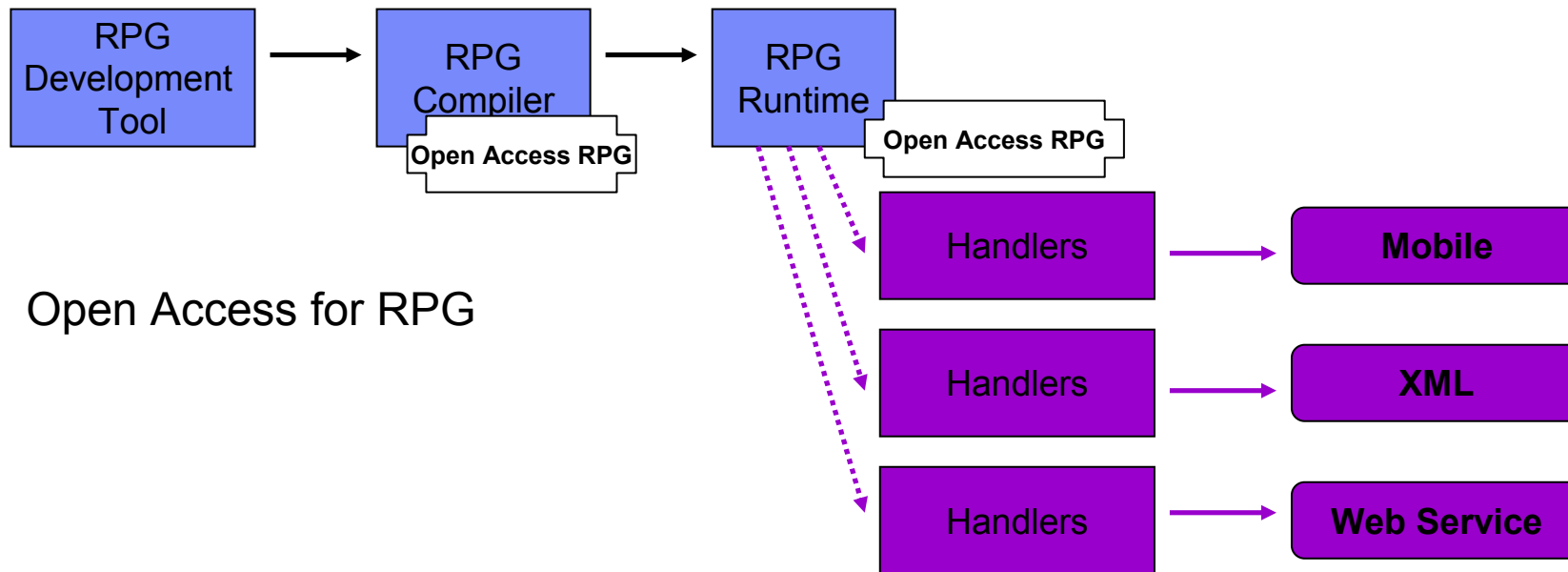


# Open Access for RPG vs HATS

## HATS



## Open Access for RPG



# 3 Parts to Open Access for RPG Solution

