February 8, 2011

# Design++ V8*i* (SELECTseries 2) Release Notes

Welcome to Bentley Design++ V8*i* (SELECTseries 2) release. This is the second follow-up release for Design++ V8*i*, which was released in October 2008 as part of Bentley's V8*i* software portfolio for infrastructure professionals.

Design++ V8*i* (SELECTseries 2) is supported on standard Windows PC platform running either 32 or 64-bit version of Windows 7, Vista (SP1), XP (SP3), or Server 2003 (SP1)/2008.

## Highlights

### 64-bit Windows Support

Design++ is now supported on 64-bit Windows 7, Vista, and XP. The main benefit of the 64-bit support is that it allows users to allocate more memory for their large Design++ models. 64-bit Windows support is also important as Autodesk supports only 64-bit AutoCAD on 64-bit Windows.

Design++ Install has been upgraded to handle both 32 and 64-bit installations; it detects the Windows version and configures Design++ accordingly.

### Unicode Support

Design++ now supports standard Unicode character set including multi-byte Asian characters. Internally, Design++ Core has been upgraded to use Unicode character representation and all socket communication have been changed to use UTF-8 for message encoding. External file encoding is now a project-specific setting.

Unicode support does not impact existing user projects not needing Unicode character set in anyway.

### Latest IDE Version Including Documentation

Integrated Development Environment (IDE) has gone through several major improvement cycles since its initial release with Design++ V8*i* (SELECTseries 1). IDE is now also properly documented in Design++ User's Manual. The previous Developer's Interface (UIP) is still available as a backup.

### Initial .NET/API Release

A new .NET/API (DNA) is introduced. DNA is a library of .NET classes that lets .NET client programs communicate with a Design++ server. This initial DNA release supports the basic communication layer (connect/disconnect, eval/send, event handling, and threading). DNA is similar to other Design++ communication APIs: C/API, COM/API, and JAVA/API.

**Support for MicroStation V8*i* (SELECTseries 2) Added**

Design$^{++}$ MicroStation V8*i* link now supports MicroStation V8*i* (SELECTseries 2).

**Support for 64-bit AutoCAD 2009, 2010, and 2011 Added**

Design$^{++}$ AutoCAD link now supports 64-bit AutoCAD 2009, 2010, and 2011.

**Supported CAD versions**

Design$^{++}$ V8*i* (SELECTseries 2) integrates with the following CAD versions.

**MicroStation:** V7 (aka J), V8 XM Edition, V8*i*, and V8*i* (SELECTseries 1 & 2)

**AutoCAD (32-bit):** 2004, 2005, 2006, 2007, 2008, 2009, 2010, and 2011

**AutoCAD (64-bit):** 2009, 2010, and 2011

## 64-bit Windows Support

- Design$^{++}$ now supports 64-bit Windows 7, Vista, and XP. The main benefit of the 64-bit support is that it allows users to allocate more memory for their large Design$^{++}$ models. 64-bit Windows support is also important as Autodesk supports only 64-bit AutoCAD on 64-bit Windows.

- Design$^{++}$ Install has been upgraded to handle both 32 and 64-bit installations; it detects the Windows version and configures Design$^{++}$ accordingly.

- Most development DLLs (C/API, COM/API, CIM) are now provided both as 32 and 64-bit versions.

- Depending on the configuration, 64-bit Design$^{++}$ installation may contain both 32 and 64-bit modules. The module breakdown is as follows:

    - 64-bit modules: Lisp images, AutoCAD link, ODBC drivers, C/API, COM/API, and CIM.

    - Platform-independent modules (no need for separate 32 vs. 64-bit versions): .NET/API and Java/API

    - 32-bit modules not yet ported: MicroStation link (64-bit version not yet available)

    - 32-bit modules: GNU Emacs and old Galaxy (now obsolete) applications (UIP, DRE, GUIB)

- On 64-bit Windows, both the 32 and 64-bit modules are installed. Design$^{++}$ is installed as a 32-bit application, but the various 64-bit modules are run by default. User can opt to run 32-bit modules (Lisp) instead by adding the following line to the top of the <d++>\bin\d++.bat file.

    ```
    set DPP64BIT=no
    ```

- On 32-bit Windows, the 64-bit modules are never installed.

## Directory Structure Reorganization

- Design$^{++}$ directory structure for system files is reorganized to support concurrent 32 and 64-bit modules. All system files are stored in either win32-i86 (32-bit) or win64-i86 (64-bit) platform subdirectory. System files are further categorized based on their usage as runtime, example, or compilation (includes and libraries) files.

| Old directory structure | New directory structure |
|---|---|
| | arx2007-interface\win64-i86\ |
| | arx2010-interface\win64-i86\ |
| arx2004-interface\win32-i86\acaddpp.lib | arx2004-interface\arx\win32-i86\acaddpp.lib |
| arx2007-interface\win32-i86\acaddpp.lib | arx2007-interface\arx\win32-i86\acaddpp.lib |
| | arx2007-interface\arx\win64-i86\acaddpp.lib |
| arx2010-interface\win64-i86\acaddpp.lib | arx2010-interface\arx\win32-i86\acaddpp.lib |
| | arx2010-interface\arx\win64-i86\acaddpp.lib |
| arx2004-interface\win32-i86\dppdefs.h | arx2004-interface\arx\include\dppdefs.h |
| arx2007-interface\win32-i86\dppdefs.h | arx2007-interface\arx\include\dppdefs.h |
| arx2007-interface\win64-i86\dppdefs.h | arx2007-interface\arx\include\dppdefs.h |
| arx2010-interface\win32-i86\dppdefs.h | arx2010-interface\arx\include\dppdefs.h |
| arx2010-interface\win64-i86\dppdefs.h | arx2010-interface\arx\include\dppdefs.h |
| | |
| arx2004-interface\arx* | arx2004-interface\Examples\dppextap* |
| arx2007-interface\arx* | arx2007-interface\Examples\dppextap* |
| arx2010-interface\arx* | arx2010-interface\Examples\dppextap* |
| | |
| | bin\win64-i86\ |
| | |
| capi*.h | capi\include*.h |
| | capi\win64-i86\ |
| | |
| cim\example\ | cim\examples\ |
| | cim\win64-i86\ |
| | cim\examples\win64-i86\example.exe |
| | |
| comapi\dppCOMapi.* | comapi\win32-i86\dppCOMapi.* |
| comapi\dppCOMserver.* | comapi\win32-i86\dppCOMserver.* |
| comapi\dppCOMserver2.* | comapi\win32-i86\dppCOMserver2.* |
| | comapi\win64-i86\dppCOMapi.* |
| | comapi\win64-i86\dppCOMserver.* |
| | |
| misc\select\ | select\win32-i86\ |
| | select\win64-i86\ |
| | |
| msj-interface\mdl* | msj-interface\examples\dppextap* |
| | |
| msxm-interface\win32-i86\msdpp.lib | msxm-interface\mdl\win32-i86\msdpp.lib |
| msxm-interface\mdl\dppextap\ | msxm-interface\examples\dppextap\ |
| msxm-interface\mdl\dppui\ | msxm-interface\examples\dppui\ |
| | |
| msv8i-interface\win32-i86\msdpp.lib | msv8i-interface\mdl\win32-i86\msdpp.lib |
| msv8i-interface\mdl\dppextap\ | msv8i-interface\examples\dppextap\ |
| msv8i-interface\mdl\dppui\ | msv8i-interface\examples\dppui\ |
| | |
| | rdb-interface\win64-i86\rdblink.exe |
| | |
| projects\geo-test\arx2004\dppextap.arx | projects\geo-test\arx2004\win32-i86\dppextap.arx |
| projects\geo-test\arx2007\dppextap.arx | projects\geo-test\arx2007\win32-i86\dppextap.arx |
| | projects\geo-test\arx2007\win64-i86\dppextap.arx |
| projects\geo-test\arx2010\dppextap.arx | projects\geo-test\arx2010\win32-i86\dppextap.arx |
| | projects\geo-test\arx2010\win64-i86\dppextap.arx |
| | |
| projects\geo-test\msj\mdl\dppextap.* | projects\geo-test\msj\win32-i86\dppextap.* |
| projects\geo-test\msxm\mdl\dppextap.* | projects\geo-test\msxm\win32-i86\dppextap.* |
| projects\geo-test\msv8i\mdl\dppextap.* | projects\geo-test\msv8i\win32-i86\dppextap.* |

## Unicode Support and Project-Specific File Encoding

- Design$^{++}$ now supports standard Unicode character set including multi-byte Asian characters. Internally, Design$^{++}$ Core has been upgraded to use Unicode character representation and all socket communication have been changed to use UTF-8 for message encoding.

  In order to save project files containing Unicode characters, the file encoding needs to also support Unicode. File encoding is now a project-specific setting. While the default file encoding is based on user's current Windows locale settings, a locale-independent UTF-8 is offered as an optional encoding for those projects that need it. UTF-8 is the standard encoding for Unicode character set and as such it facilitates project sharing between distributed teams across various Windows locales.

  Unicode support does not impact existing user projects not needing Unicode character set in anyway. Design$^{++}$ will continue to run existing projects using the default file encoding based on user's current Windows locale settings. This is the encoding the projects were created with in the first place.

## Installation

- Java JRE is no longer installed automatically. Instead, users need to install it themselves if they want to use Java/API or run the old Java-based RelationBrowser.

- Franz ACL Runtime bundle (files.bu) file and locales directory are now included with the delivery. This enables the use of non US-EN locales in Lisp Images. The bundle includes also some other modules not included in the Design$^{++}$ image by default.

## Design$^{++}$ Core

- New Lisp binary file extensions are 32w82fsl (32-bit Lisp compiler) and 64w82fsl (64-bit Lisp compiler).

- Finnish language versions of design rule macros are declared obsolete.

- Design rule compilation is fixed not to trigger the change propagation until the rule is properly stored. This fixes the problem where design rule lost its formatting if the subsequent change propagation was cancelled.

- Tracing API communication (TRACE-API) between Telnet Server and IDE is fixed to handle the PRINT-FULL-MESSAGES-P option correctly.

- The Emacs interface initialization problem that sometimes prevented Design$^{++}$ from starting when run with Emacs is fixed. It seemed to be a timing problem occurring more frequently on newer and faster machines.

- The loading of a startup project (DPPSTARTUPPROJECT) and executing a startup function (DPPSTARTUPFUNCTION) during Design$^{++}$ startup are fixed to have their

output directed to Emacs (when run with Emacs) and finally return control back to Emacs by restoring its prompt.

- The error message for function START-EXTERNAL-SERVER is improved to include also the shell command that was used to attempt to start the server.

- Function CURRENT-RELATIONS is modified to support uni-directional relations by not assuming that a relation is its own *opposite-relation* by default.

- Functions DPP-GET-VALUE and DPP-REQUEST/REMOVE-ATTRIBUTE-VALUE are modified to return only the local or inherited attribute value for classes (and instances) that are not in the current model. Determining the attribute value by a design rule, retrieving the value from an external data source, or prompting the value from the user are reserved for component instances (and hidden components) in the current model only.

- Function DPP-CREATE-PROJECT has a new argument FILE-ENCODING for specifying the default external format.

- Functions DPP-RELATE and DPP-UNRELATE have a new optional argument UPDATE-GUI-P.

## Integrated Development Environment (IDE)

- Integrated Development Environment (IDE) has gone through several major improvement cycles since its initial release with Design$^{++}$ V8*i* (SELECTseries 1). For example,

    - IDE now supports standard Unicode character set including multi-byte Asian characters.

    - CAD startup is modified to start CAD only if the CAD link is enabled. This is a user-settable Design$^{++}$ preference.

    - New Design$^{++}$ Preferences and Project/Library/Model Properties dialogs are added.

    - Project Navigator now supports basic file operations for Functions, Rules, Structures, and External Data nodes.

    - New Query Analyzer is introduced.

    - General window layout is improved.

    - Menus, dialogs, prompts, and icons are revisited for consistency.

    - Class, Model, and Relation Browsers are optimized to handle large models (1,000+ components).

    - IntelliSense information retrieval for Rule Editor is optimized.

- IDE is now properly documented in Design$^{++}$ User's Manual.

- The previous Developer's Interface (UIP) is still available as a backup.

## C/API

- Internally the communication with Lisp is now using UTF8 as the data type. This is implemented by replacing the old EUC data handling with UTF8 data handling and setting the UTF8 as the default data type.

- The default output character type, dppPortChar, is still 'char'. To use Unicode (wchar_t), add a call to dppPortInitializeWchar before other C/API calls and add a compiler preprocessor definition 'dppPortChar=wchar_t' to the project settings. Optionally, change the Visual Studio project setting 'Character Set' to 'Use Unicode Character Set' and/or add preprocessor defines UNICODE/_UNICODE.

- The old EUC term in names has been changed to term UTF, for example,

    - `dppCommDATA_EUC -> dppCommDATA_UTF`
    - `dppPortCHAR_TO_EUC -> dppPortCHAR_TO_UTF`
    - `dppPortEUC_TO_CHAR -> dppPortUTF_TO_CHAR`

- The Portability Manager has been modified as follows.

    - dppPortInitializeWchar can be used to specify that wchar_t is the output type.
    - Previously all Portability Manager functions had to be registered. Now all the functions have defaults, and functions can be registered individually as needed.
    - dppPortALLOC is renamed to dppPortMALLOC.
    - A new function dppPortCALLOC for allocating arrays is introduced.
    - dppPortCHAR_TO_UTF (renamed from dppPortCHAR_TO_EUC) now converts from client string to utf8.
    - dppPortUTF_TO_CHAR (renamed from dppPortEUC_TO_CHAR) now converts from utf8 to client character set.

- C/API include files are now stored in <d++>\capi\include\ instead of <d++>\capi\

- 64-bit Windows binaries are now supported. In order to compile for 64-bit Windows, you first need to convert your project settings to compile with 64-bit compiler. Also, change the C/API library path from <d++>\capi\win32-i86\ to <d++>\capi\win64-i86\

- New functions are introduced: dppProjectExists, dppLibraryExists, dppModelExists, dppProjectSetProjectsPath

- 'const' argument qualifiers are added to some functions.

- dppStringArrayFlatten and dppStringArrayFlattenWithSeparator used to return strings with extra ending space characters. This issue is now fixed.

- Starting with VC++ 2005 Single-Threaded Runtime libraries are no longer supported. Related C/API libraries are also removed.

- The problem, where VC++ compiler settings didn't always correctly match the library name, is now fixed.

## COM/API

- Communication with Lisp now supports standard Unicode character set including multi-byte Asian characters.

- Both 32 and 64-bit versions are now delivered

## .NET/API

- A new Design$^{++}$ .NET Application Programming Interface, .NET/API or DNA for short, is introduced. This initial DNA release supports the basic communication layer (connect/disconnect, eval/send, event handling, and threading).

  DNA is a library of .NET classes that lets .NET client programs communicate with a Design$^{++}$ server. DNA is built on top of sockets and it allows communication over the network using the TCP/IP protocol. DNA is similar to other Design$^{++}$ communication API's: C/API, COM/API, and JAVA/API.

  Developing applications with the DNA is based on a client-server model where a Design$^{++}$ server waits to be contacted by a client application so that it can provide some service for the client. A client sends messages across the network, or locally, to the server requesting service of some form. DNA manual includes an example of a simple C# console application connecting with a Design$^{++}$ server.

## Starting Design$^{++}$ from CAD

- A new document "Starting Design$^{++}$ from cad.pdf" is now included in <d++>\documentation\ directory.

## CAD Integration Manager (CIM)

- 64-bit Windows binaries are now supported. In order to compile for 64-bit Windows, you first need to convert your project settings to compile with 64-bit compiler. Also,

change C/API library path from <d++>\capi\win32-i86\ to <d++>\capi\win64-i86\ and CIM library path from <d++>\cim\win32-i86\ to <d++>\cim\win64-i86\

- Communication with Lisp now supports standard Unicode character set including multi-byte Asian characters.
- Examples directory is renamed from <d++>\cim\example\ to <d++>\cim\examples\
- Starting with VC++ 2005 Single-Threaded Runtime libraries are no longer supported. Related CIM libraries are also removed.

## MicroStation Link

- Support for MicroStation V8*i* (SELECTseries 2) is added.
- Unicode support is added to MicroStation XM and V8*i* links.
    - Links are converted to use 'MSWChars' instead of 'char' wherever possible.
    - The native dppextap MDL API is also converted to use string type 'MSWChar' instead of 'char'.
    - Some features behave according to the current Windows Locale.
        - File access. For example, in order to use Japanese file names, locale has to be Japanese.
        - MLText field in Attribute Edit dialog in DPPUI example works with multi-byte strings supported by the current locale.
        - Other features, like color and font names.
    - dppstart utility does not support Unicode text.
    - Note that MicroStation/J link does not support Unicode.
- MicroStation link directory structure is modified.
    - dppextap example is now in <d++>\<ms-interface>\examples\dppextap\
    - dppui example is now in <d++>\<ms-interface>\examples\dppui\
    - msdpp.lib is now in <d++>\<ms-interface>\mdl\win32-i86\
- Function dppExtapGetAttributeValuePairs returns now MDL data type ListModel instead of StringList.
- An error, where only the first specified dppextap was loaded, is now fixed.

## AutoCAD Link

- Support for 64-bit AutoCAD versions 2009, 2010, and 2011 is added. Note that 64-bit AutoCAD 2008 is not supported.
- Unicode support is improved.
    - Communication with Lisp now supports standard Unicode character set including multi-byte Asian characters.
    - Some features behave according to the current Windows Locale, like AutoLISP.
- dppstart utility is not usable from VBA in 64-bit AutoCAD. Also, dppstart utility does not support Unicode text.
- AutoCAD link directory structure is modified.
    - ARX include files are now in <d++>\<arx-interface>\arx\include\
    - acaddpp.lib 32-bit library file is moved to <d++>\<arx-interface>\arx\win32-i86\
    - acaddpp.lib 64-bit library file is in <d++>\<arx-interface>\arx\win64-i86\
    - dppextap example is now in <d++>\<arx-interface>\examples\dppextap\
- External application search is modified to check <project>\<cad>\<sysdir> before <project>\<cad>

## Database Link

- Communication with Lisp now supports standard Unicode character set including multi-byte Asian characters.
- Support is added for 64-bit ODBC drivers. By default, 32-bit ODBC drivers are used with 32-bit Lisp, and 64-bit ODBC drivers with 64-bit Lisp. This behavior can be changed with argument :ARCHITECTURE in functions DPP-DB-SETTINGS, DPP-DB-CONNECT, and DPP-DB-WITH-CONNECTION-SETTINGS.

  For example, when running 64-bit Design$^{++}$ on 64-bit Windows with 32-bit Office, 32-bit ODBC drivers need to be used:

```
(dpp-db-settings :connection "dpp_test" :architecture :win32)
```

## Emacs Interface

- Emacs version is updated to GNU Emacs 23.2

- The Emacs interface initialization problem that sometimes prevented Design$^{++}$ from starting when run with Emacs is fixed. It seemed to be a timing problem occurring more frequently on newer and faster machines.

- Deleting files in Emacs is modified to send the files to Recycle Bin. This is a setting in Emacs initialization file dpp.el.

## Java/API

- Communication with Lisp now supports standard Unicode character set including multi-byte Asian characters.

- Function DPP-JAVA-OPEN-DIALOG is fixed to return an error string in case of an error.

- Java JRE is no longer installed automatically. Instead, users need to install it themselves if they want to use Java/API or run the old Java-based RelationBrowser.

## Licensing

- SELECT License Client is upgraded to the latest version 08.11.07.45.

- SELECT License Client files are now in <d++>\select\win32-i86\ and <d++>\select\win64-i86\

## Miscellaneous

- Example project Table is updated to demonstrate model creation using the :INIT macro.

- Support for Acrobat Reader X is added for online documentation access.

- Design$^{++}$ startup is modified to show an error dialog and keep the console window executing d++.bat file open in case the startup fails. This provides user some information about the startup problem.

- Most of the C/C++ code is modified to use _WIN32 in ifdef's instead of WIN32, WINDOWS, or _WINDOWS as it is always defined by VC compiler.

- Some example VC projects are modified to use path like below for compilation output files.

    Outputs\<ProjectName>\<Platform>\<CompilerVersion>\<Configuration>\

  For example, AutoCAD link project dppextap for AutoCAD 2010 with 64-bit VC 9 and 'Release' Configuration uses the following path.

    Outputs\dppextap90\x64\VC90\Release\