March 12, 2007

Design++ 6.1 Release Notes for Windows 2000/XP/2003

We are pleased to announce the release of Design++ 6.1 for Windows 2000/XP/Server 2003. This is an incremental release for Design++ 6.0 with AutoCAD 2007 support as the most important added feature.

HIGHLIGHTS

NEW LICENSE MANAGER

The new license manager version fixes, among other things, the infamous problem of repeated loss of temporary licenses, especially network licenses.

Unfortunately, the new license manager is not compatible with previous license manager versions. A license manager upgrade is available for Design++ 6.0 and 6.0B2 to allow them to share a common license (pool) with Design++ 6.1. If you are interested in the upgrade, please contact your Design++ representative.

ENHANCED INSTALLATION

Design++ 6.1 supports silent installations through MacroVision InstallShield Silent installation capability. With silent installation, there is no need for the user to monitor the setup and provide input via dialog boxes. A silent setup runs on its own without any user intervention.

Also, the normal installation is now customizable through an init file, which can be used to specify default values for various installation settings. User prompts can also be blocked.

NEW FRANZ ACL VERSION

Design++ 6.1 is deployed with the latest Franz ACL 8.0.

SUPPORTED CAD VERSIONS

With the AutoCAD 2007 support added, Design++ 6.1 integrates with the following CAD versions.

AutoCAD: 2000, 2000i, 2002, 2004, 2005, 2006, and 2007.
MicroStation: V7, V8 2004 Edition, and V8 XM Edition.
Visio: 2000, 2000 SR1, 2002, 2003, and 2007.

WINDOWS VISTA SUPPORT

Windows Vista is not yet supported, but preliminary testing indicates that if installed and used with the same user account, Design++ does run on Vista.

DESIGN++ FUNCTIONS (LISP/API)

* New optional argument DELETE-ALL-SUBCLASSES-P added for Design++ function DPP-DELETE-CLASS. The new argument determines whether to

```
   delete all subclasses recursively. Note that subclasses are still
   only deleted within the CLASS-KB-NAME (library).

   ;|| DPP-DELETE-CLASS (class-name class-kb-name
   ;||                  &optional delete-direct-subclasses-p
   ;||                            delete-all-subclasses-p)
   ;|| PURPOSE:
   ;||  Deletes an existing class and optionally its subclasses.
   ;|| ARGUMENTS:
   ;||  class-name:
   ;||   Name of the class to be deleted (symbol).
   ;||  class-kb-name:
   ;||   Name of the kb of the class to be deleted (symbol).
   ;||  Optional:
   ;||   delete-direct-subclasses-p:
   ;||    Whether to delete direct subclasses. T for yes or NIL (default)
   ;||    for no.
   ;||   delete-all-subclasses-p:
   ;||    Whether to delete all subclasses recursively within the kb. T
   ;||    for yes or NIL (default) for no.
   ;|| RETURNS:
   ;||  T if class is deleted, otherwise NIL.
   ;|| EXAMPLE:
   ;||  (dpp-delete-class 'testclass 'reports)
   ;||  ==> T
   ;||

 * Design++ Function DPP-COPY-CLASS revisited.

   1. Arguments renamed to be more descriptive.

   2. Handling of NEW-SUPERCLASSES modified to accept both classes
      and class references.

   3. Documentation updated and notes added regarding the handling of
      local data for inherited attributes and design rules. See below.

      ;|| DPP-COPY-CLASS (class library new-name
      ;||                &optional new-library copy-subclasses-p
      ;||                          new-superclasses)
      ;|| PURPOSE:
      ;||  Makes a copy of an existing class
      ;|| ARGUMENTS:
      ;||  class:
      ;||   Class to be copied (class-or-ref)
      ;||  library:
      ;||   Library of the class to be copied (symbol)
      ;||  new-name:
      ;||   Name for the new class (symbol)
      ;||  Optional:
      ;||   new-library:
      ;||    Library for the new class. Defaults to the library of the class
      ;||    to be copied (symbol).
      ;||   copy-subclasses-p:
      ;||    Whether to copy subclasses also (T or NIL (default)).
      ;||   new-superclasses:
      ;||    List of superclasses for the new class. Defaults to the
      ;||    superclasses of the class to be copied (list of class-or-refs).
      ;|| RETURNS:
      ;||  New class if the copy succeeded, otherwise NIL.
      ;|| EXAMPLE:
      ;||  (dpp-copy-class 'pump 'plant 'my-pump)
      ;||  ==> #<FRAME MY-PUMP PLANT>
      ;||
      ;|| NOTES:
      ;||  1. If the class to be copied has local data for attributes that
      ;||  it has inherited, you need to make sure that those attributes are
```

```
     ;||  also inherited for the copied class. If not, dpp-copy-class will
     ;||  fail. To make sure all attributes are properly inherited, you
     ;||  need to add all existing superclasses of the class to be copied
     ;||  to the list of NEW-SUPERCLASSES. This is handled by default when
     ;||  copying within a single library, but not when copying from one
     ;||  library to another.
     ;||
     ;||  2. Design rules are copied as such without modifying every CLASS
     ;||  symbol with NEW-NAME symbol in the rule body. Still, the copied
     ;||  rules will function properly in the context of the copied class
     ;||  as long as all the self references are done using SELF, like (:?
     ;||  self), instead of (:? CLASS).
     ;||
     ;||  3. Compiled rules are also copied as such. This means that unless
     ;||  the copied rules are renamed locally from (:! CLASS <attr>) to
     ;||  (:! NEW-NAME <attr>) and then recompiled, the rules that are
     ;||  executed are actually those of the original CLASS. Thus, for as
     ;||  long as you have not modified a copied rule, you are running the
     ;||  original rule. Likewise, if the original rule is modified, then
     ;||  it will "inherit" also to the copied class. Remember that if you
     ;||  do modify a copied rule, you need to make sure that you also
     ;||  rename it to (:! NEW-NAME <attr>).
     ;||

-------------
DESIGN++ CORE
-------------

* Compiler settings revisited.

  1. Development mode compiler settings modified to maximize safety
     and debugability; optimizing speed and space is secondary.
     Production mode compiler settings optimize speed as before.
     Thus, the new mode settings are as follows.

     DEVELOPMENT mode compiler settings:
     (proclaim '(optimize (safety 3) (space 0) (speed 0) (debug 3)))

     PRODUCTION mode compiler settings:
     (proclaim '(optimize (safety 1) (space 0) (speed 3) (debug 0)))

     Compiler mode is set either through 'Design++ Preferences'
     dialog or with Design++ function DPP-SET-COMPILER-MODE.

  2. New Design++ Function DPP-GET-COMPILER-MODE introduced.

     ;|| DPP-GET-COMPILER-MODE
     ;|| PURPOSE:
     ;||  Returns current Design++ compiler mode and compiler
     ;||  optimization quality settings.
     ;|| ARGUMENTS:
     ;|| RETURNS:
     ;||  Two (multiple) values: current compiler mode
     ;||  (:PRODUCTION, :DEVELOPMENT, or :CUSTOM) and the actual
     ;||  compiler optimization quality settings (SAFETY, SPACE,
     ;||  SPEED, and DEBUG)
     ;|| EXAMPLE:
     ;||  (dpp-get-compiler-mode)
     ;||  ==> :DEVELOPMENT
     ;||  ==> ((SAFETY 3) (SPACE 0) (SPEED 0) (DEBUG 3))
     ;||

* A Franz patch to fix a problem where the compiler failed to
  recover from a simple syntax error causing the whole Design++
  session to hang and eventually crash. The actual problem was
  caused by an error in compiler error message printing.
```

DYNAMIC CONFIGURATION ENHANCEMENTS

* FrOBS-level PUT-VALUE and GET-VALUE functions and all of their derivatives modified to use equal instead equalp to determine if an attribute value has changed. As a result, attribute value changes like

  ```
  #\A      -> #\a
  "abc"    -> "ABC"
  '("abc") -> '("ABC")
  ```

  are now properly detected.

INSTALLATION

* Checking for environment variable HOME improved. Design++ requires that HOME is set and that it points to a valid directory, typically user's home directory under "C:\Documents and Settings\". Environment variables can be set by clicking the 'Environment Variables' button from ControlPanel->System->Advanced tab.

* Silent installations are supported through MacroVision InstallShield Silent installation capability. With silent installation, there is no need for the user to monitor the setup and provide input via dialog boxes. A silent setup runs on its own without any user intervention.

* The normal installation is now customizable through an init file, which can be used to specify default values for various installation settings. User prompts can also be blocked.

LICENSE MANAGER

* The new license manager version 6.5.25 fixes, among other things, 2 critical licensing problems, namely

  - Repeated loss of temporary licenses, especially network licenses.
  - Loss of license if the license service is not running when the user tries to start Design++.

  Note that older license manager versions are not fully compatible with version 6.5.25. If you are planing to run an older Design++ version on the same workstation with Design++ 6.1 or to share a license server with Design++ 6.1, it's critical that you upgrade the older Design++ version to use the same license manager version 6.5.25.

  A license manager upgrade is available for Design++ 6.0 and 6.0B2 to allow them to share a common license (pool) with Design++ 6.1. If you are interested in the upgrade, please contact your Design++ representative. Note that upgrading to the new license manager version does NOT require your Design++ license to be reauthorized.

C/API

* Portability Manager revisited as part of the AutoCAD 2007 Unicode (wchar) integration. A TCHAR example is available in <dpp>\capi\example\port

* Added a new function dppCommGetDebugEnabled, which returns the status of CAPI communication debug output setting.

```
     dppBool dppCommGetDebugEnabled(void)

* Added a new Portability Manager interface specification function
  dppPortFTOA, which converts a double to a string.

     void ftoaFunc(dppchar *str, double d);

* Added new Portability Manager interface specification function
  dppPortSPRINTF, which formats the parameters to a buffer, like
  sprintf.

     void sprintfFunc(dppChar *buf, dppChar *fmt, ...);

------
COMAPI
------

* Modified to report event message handling errors with a dialog box
  instead of just exiting.

-----------------------------
CAD INTEGRATION MANAGER (CIM)
-----------------------------

* The use C/API Portability Manager revisited as part of the AutoCAD
  2007 Unicode (wchar) integration.

----------------
AUTOCAD/ARX LINK
----------------

* Starting with AutoCAD 2006, symbol attributes have property
  lockPositionInBlock, which controls the attribute location behavior
  when symbol is modified. Primitive symbol-with-attributes is fixed
  to handle this property correctly.

* AutoCAD 2007 support added.

* The external application (dppextap) example modified to use Unicode
  (TCHAR) for AutoCAD 2007.

* Design++ submenu files modified to include toolbar, item icons, and
  item help strings.

----------
VISIO LINK
----------

* Error messages revisited to be more informative.

---------------
EMACS INTERFACE
---------------

* The :ED Lisp debugger command in Emacs fixed to bring up also
  source code for a CLOS method associated with the current frame in
  the call stack. The command used to work only for functions and
  design rules.

----------------------------
USER INTERFACE PROGRAM (UIP)
----------------------------

* The 'Release Notes' entry in the Help menu fixed.

----
```

```
MISC
----

* The new Franz ACL 8.0 fixes the problem associated with Windows Data
  Execution Prevention Option (DEP), which, if turned on, caused
  Design++ to crash during startup.

=====
```