PDK Compatibility Documentation, Version 1004

Here is a simplified summary of what a design kit developer needs to do.

Installation steps for a PDK.

1.) Put the attached zipped directory in the following location:

```
"<design_kit_root>/de/ael/compat"
```

So the dk_compatibility.atf and dkcompatLoadFile_***.atf files are all found under this directory:

```
"<design_kit_root>/de/ael/compat/dk_compatibility.atf"
"<design_kit_root>/de/ael/compat/dkcompatLoadFile_350.atf"
"<design_kit_root>/de/ael/compat/dkcompatLoadFile_370.atf"
"<design_kit_root>/de/ael/compat/dkcompatLoadFile_COMMON.atf"
```

2.) Modify the design kit's boot.ael file to have the following load line:

```
// designKitRecord[1] holds path to the design kit.
load( strcat(designKitRecord[1],  "/de/ael/compat/dk_compatibility") );
```

3.) Now the design kit can use the following functions at this time in releases prior to when they are actually available:

- is_pdeoa_mode() – was originally made available in ADS 2009

- dkcompat_flatten_layout_into_current() – available currently only via this compatibility method for use in design kits only.

- netlist_instance_cb() – originally made available in ADS 2011. A netlist format string is preferred for use over this function since the format string syntax is the most backwards compatible way of netlisting components in all ADS versions and is most efficient. This function is limited in what is netlisted, it won't netlist repeated parameters and assumes that parameter values have a left and right side "<parameter name> = <param value>". The types of parameter forms is somewhat limited in the compatibility version of this function. It has the same limitation on what parameter values can be properly netlisted as generic_netlist_cb() had.

- de_is_converting_dsn_pdk()—a backward compatibility function that gets loaded into ADS 2009 Update 1 or prior releases, and could be used in PDK ael code that is expected to run in both ADS 2009 Update 1 or prior releases as well as in ADS 2011. This function was first implemented in ADS 2011.01 to support a special need for converting two PDKs one of which was primary and the other secondary, and that the secondary PDK conversion had a dependency on a certain AEL in the primary one. This function could be called from the secondary PDK's ael where it relied on this function to detect that the primary PDK to be available, and therefore retrieve the necessary information from the primary PDK. This function always returns FALSE when used in ADS 2009 Update 1 or prior releases.

Documentation for new functions:

```
// Function:  is_pdeoa_mode()
// Description:
// This function returns TRUE if the current version of ADS, is ADS 2011
// or a version of ADS newer than ADS 2011 (does not yet exist).
// Otherwise if the current version of ADS, is ADS 2009 Update 1 or any ADS
// version before ADS 2009 Update 1 this function will return FALSE.
//
// Syntax:  decl isADSOA = is_pdeoa_mode();
//
// Example:
//   if (is_pdeoa_mode())
//   {
//       db_select(dgH, TRUE);
//   }
//   else
//   {
//       db_set_dg_attribute(dgH, DG_SELECT, TRUE);
//   }


// Function:  de_is_converting_dsn_pdk()
// Description: this function, when loaded into ADS 2009 Update 1 or prior
// releases via the backward compatibility ael file, will always return
// FALSE.
//
// Example:
//   if (de_is_converting_dsn_pdk())
//   {
//       decl tempDesignKitRecord = designKitRecord;
//       designKitRecord = list("PDK1","C:/users/PDK1/","","v1");
//       load("C:/users/PDK1/ael/stmenu.ael");
//       designKitRecord = tempDesignKitRecord;
//   }
```

```
--


// Function:  dkcompat_flatten_layout_into_current()
// Description:
// This function copies the fixed artwork from one design to use for itself
// in the current design.
// * For use in artwork macros.
// * When called the current design should be a current layout design view.
//
// Syntax:
// dkcompat_flatten_layout_into_current(sourceLibName,
//    sourceCellName, sourceViewName, [, selectedOnly[, includePins]]);
// where
//   sourceLibName:  is ignored by ADS 2009 Update 1 and earlier. In ADS 2011
//     and newer releases, this is the source layout design's library name
//     to get the fixed artwork from.
//   sourceCellName:  is the source layout design name for ADS 2009 Update 1
//     and earlier. In ADS 2011 and newer releases, this is the source
//     layout design's cell name to get the fixed artwork from.
//   sourceViewName:  is ignored by ADS 2009 Update 1 and earlier. In ADS
//      2011 and newer releases, this is the source layout design's view
//      name to get the fixed artwork from.
//   optional:  selectedOnly - used only in ADS 2011 and newer releases.
//      Ignored by ADS 2009 Update 1 and earlier releases.
//      Default is FALSE if not provided.
//      If TRUE, will only copy those instances that are selected when
//      flattening.
//   optional:  includePins - used only in ADS 2011 and newer releases.
//      Ignored by ADS 2009 Update 1 and earlier releases.
```

```
//      Default is FALSE if not provided.
//      If TRUE, will copy the pins when flattening.
//
// Notes:
// This one function below can be used in a design kit for any version of ADS
// that has LMSArtworkFromGeom defined.
// Made one compatible function because:
// For ADS 2010 and newer versions,
//  de_flatten_all(sourceContext, targetContext,[selectedOnly,[includePins]])
// is recommended.
// For all ADS 2009 Update 1 and earlier DSN versions:
//  LMSArtworkFromGeom(geom, LMSLibName);
// is recommended.
//
// Example:
//   decl libName = "mylibrary";
//   decl cellName = "mydsn";
//   decl viewName = "layout";
//
//   dkcompat_flatten_layout_into_current(
//       libName, cellName, viewName, FALSE, TRUE);
```

--

```
// Function:  netlist_instance_cb()
// Description:
// Simple netlist callback function that will netlist all of the nodes,
// pins, and parameters that are listed as being netlisted.  The function
// will output an instance using gemini format.
// To use a parameter for the component name, the data field of the callback
// should be set to be the parameter name to use.
// Syntax:
//   netlist_instance_cb( cbP, modelParam, instH,
//                        iterationInstName, iterationPinList )
// where
//   cbP:  is ignored.
//   modelParam:  is the callback data field where if it is specified
//      and not NULL, will cause the netlist_instance_cb() function to
//      search for the given parameter name and if found it will use
//      its value as the component name that is netlisted. If the modelParam
//      is not specified (is NULL or "") or not found, then the item
//      definition's netlistData field if specified. If the netlistData
//      field is not specified (is NULL). Then the item definition's name
//      field is used as the netlisted component name.
//   instH: the handle to the instance object that is being netlisted.
//   iterationInstName: the instance name of the particular instance
//      iteration that is being netlisted. e.g. "R1<4>" or (for a non-
//      iterated instance) "R2"
//   iterationPinList: a list of 4-tuplets, one for each pin.
//      Each tuplet is:
//          list( pinName, pinNumber, nodeName, nodeNumber )
//      The nodeName is what is netlisted if specified.
```

```
//
// Notes/Limitations:
// A.) Limitations on the netlisting of the parameter values.
// In ADS 2009 Update 1 and earlier, the netlisted value is retrieved
// by calling getInstanceValueFromName() similar to generic_netlist_cb()
// In ADS 2011 the formset evaluator of the netlist will be used to retrieve
// the netlisted value by calling the function
//  db_param_iter_get_netlist_value().
//
// Each parameter should be output with the format [name]=[value].
// This code assumes that the parameter will have a single output value for
// each parameter. This code assumes that the parameter value will not be
// a repeated value.
//
// Example:
//
// create_item("My_Component", "My Component Descr", "myComp", 0, NULL, NULL,
//     standard_dialog, NULL, "", "My_Model_1", ComponentAnnotFmt,
//     "SYM_My_Component", macro_artwork, "My_Component_Artwork",
//      ITEM_PRIMITIVE_EX,
//      list
//      (
//          dm_create_cb
//          (
//              ITEM_NETLIST_CB,
//              "netlist_instance_cb",
//              "ModelName",
//              TRUE
//          )
//      ),
//      create_parm("w", "Width", 0, "StdFormSet", 5,
//          prm("StdForm",".5 um")),
//      create_parm("l", "Length", 0, "StdFormSet", 5,
//          prm("StdForm",".5 um")),
//      create_parm("ModelName", "Specify Simulator Model Name", 0,
//          "StdFormSet",-1, prm("StdForm","My_Model_2")),
// );
//
```