Autodesk University 2007

Creating Your Own Vertical Application through AutoCAD® OEM

Fernando P. Malard – OFCDesk, LLC.

CP405-1 In this session, you'll learn how to create, build, and support AutoCAD OEM solutions. You'll see detailed steps for creating an AutoCAD OEM application, and how to add features and specific application modules using Visual LISP, VBA, .NET and ObjectARX programming. At the end of the session, users will know the potential of their products and how to create their own AutoCAD verticals through AutoCAD OEM.

Key Topics:

- Basic differences between AutoCAD and AutoCAD OEM;
- How to design an OEM solution;
- Adding VisualLISP, VBA, .NET, and ObjectARX modules;
- Building and testing your solution;
- How to deploy, install and update your solution.

Target Audience:

AutoCAD power and advanced users

About the Speaker:

Fernando is a senior developer for OFCDesk L.L.C. He has been an ADN member since 1996, and specializes in AutoCAD, C++, ObjectARX, MFC, .NET, SQL, and Oracle. He is a teacher with experience in software development, ERP solutions, CAD solutions, and system integration. Fernando is an AUGI member (and serves on the AUGI board, Brazil), an Autodesk beta tester, and holds a Master's degree in Structural Engineering from the Federal University of Minas Gerais, Brazil.

Email: fernando.malard@ofcdesk.com (business) / fpmalard@yahoo.com.br (personal)

Introduction

AutoCAD® OEM (**Original equipment manufacturer**) is a special version of AutoCAD created to allow other companies to build their own branded products (for internal or external use) on the top of AutoCAD product. In this scenario AutoCAD OEM really becomes a powerful CAD platform providing a great variety of incomparable tools.

At the beginning, AutoCAD OEM was primarily used by Autodesk® itself to create its own AutoCAD-based products like **Architectural Desktop**®, **Autodesk MAP**®, etc. These products can take full advantage of AutoCAD product and add valuable industry specific features that complete these products perfectly.

There are a lot of advantages on this approach because it separates the generic CAD platform from the specific industries technologies and processes. Once the main AutoCAD platform is updated all the rest of OEM branded products will receive this update automatically on their next versions.

After using this technology for their own products, Autodesk has decided to open and offer this technology to other companies through AutoCAD OEM product. This strategy reinforces the Autodesk OEM technology from the Autodesk side and allows companies to create their custom branded products based on AutoCAD but exposing only the desired features and commands they need. There are a lot of reasons to use AutoCAD OEM solutions such as:

- Deliver a product that requires **CAD functionality** but is not targeted to traditional CAD users;
- Build products that can both **read** and **create DWG™ files** that are natively compatible with AutoCAD DWG files;
- Create host applications that **run inside a child window** or through a **web page**;
- Provide an **AutoCAD-based platform** that cannot be customized or extended by end-users;
- Replace a legacy CAD system and take advantage of new **AutoCAD software technologies**.

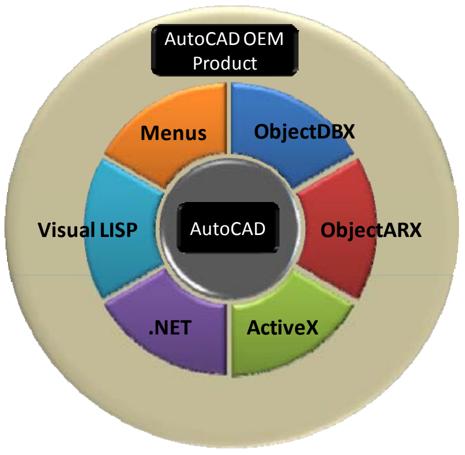


Figure 1 – AutoCAD OEM Product structure

The AutoCAD OEM product structure will be composed by AutoCAD core, AutoCAD programming interfaces and the Company features at the border layer (Figure 1). Each interface will have full access to AutoCAD features and can complete your product with industry specific features and AutoCAD cannot provide.

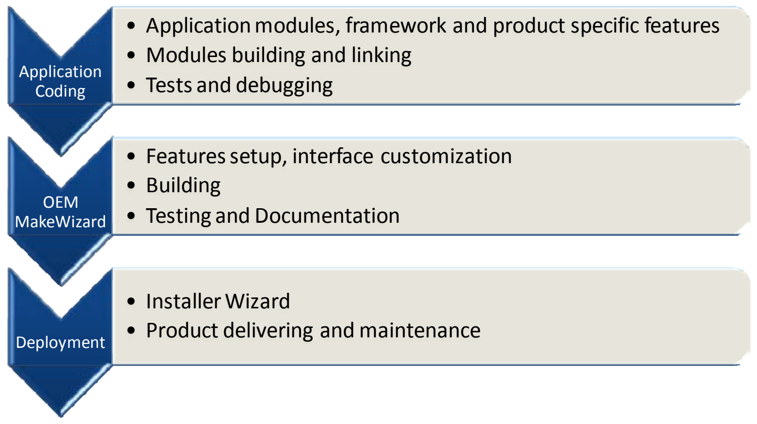

Application Coding
• Application modules, framework and product specific features
• Modules building and linking
• Tests and debugging
OEM MakeWizard
• Features setup, interface customization
• Building
• Testing and Documentation
Deployment
• Installer Wizard
• Product delivering and maintenance

Figure 2 – AutoCAD OEM Product cycle

The development process of an AutoCAD OEM product is basically divided into 3 basic steps (Figure 2). The first step is about coding, building and testing the application. If necessary this step needs to be done both inside AutoCAD OEM and AutoCAD products. The second step is about building the OEM product itself specifying its features, interfaces and performing a final test. The third and last step is the installation building and distribution.

AutoCAD OEM has the same **DWG core technology** of AutoCAD and due that it will be fully compatible with DWG files natively. Drawings created or edited inside AutoCAD OEM can be normally manipulated inside AutoCAD and any other AutoCAD-based product. If your OEM product implements custom entities you will need to provide proper Object Enablers to be used outside your AutoCAD OEM.

Differences between AutoCAD and AutoCAD OEM

There are some differences between the AutoCAD and AutoCAD OEM that may affect your applications. These differences are listed below:

1) **Commands and System Variables**:
   a. Unsupported: There are some unsupported system variables on AutoCAD OEM: **ACADVER**, ACADLSPACDOC, ACADPREFIX, LOGINNAME, MENUCNTL, POPUPS, SCREENBOXES and SCREENMODE;
   b. Specific: There are specific commands inside AutoCAD OEM: ..SYSSTATUS, LISTCOMMANDS and PKFSTGROUP. Further, there are some specific system variables: BANNER, CLIPBOARD, EXEDIR, **PROGRAM**, **PRODUCT** and **VERSION** (this is equivalent to the ACADVER system variable).
2) **User Interface**:
   a. The **text window** (command line) can be disabled to avoid text commands. You will need to disable related commands and **F2** key;
   b. AutoCAD OEM **supports menu customization** through CUI commands allowing multiple menus as AutoCAD supports;
   c. **Toolbars** can also be customized through **CUI** commands. Changes will be stored at the customization file and at Windows Registry.
3) **ObjectARX modules**:
   a. All ObjectARX and ObjectDBX modules need to be recompiled and linked with specific **AutoCAD OEM ObjectARX SDK**. These modules will be bound (**stamped**) to the main AutoCAD OEM product module and will work only within this EXE module. These modules

can be loaded using the standard procedures but it is highly recommended to use the specific AutoCAD OEM loading options like "**Required Modules**";

b. You may want to disable the Drawing Save feature of your AutoCAD OEM product. This can be done from inside your ObjectARX modules using the **acedDisableDbMod**() global function.

4) **<u>Visual LISP</u>**:

a. **Visual LISP IDE** is available to AutoCAD OEM allowing you to build and test your specific Visual LISP modules to be packed into your OEM product;

b. Specifically, although bound FAS files are supported in AutoCAD OEM, you **cannot create or use VLX files** (Visual LISP packed applications). Moreover, you **cannot use Visual LISP functions related to ActiveX**®, including **vl-load-com**, **vlax-*** and **vlr-*** functions. If you use these methods your Visual LISP module will not work with your OEM product;

c. **Only compiled AutoLISP modules are supported** and need to be bound (stamped) with a specific host module (your OEM product EXE main module);

d. **End users cannot enter AutoLISP expressions** through the command line, menu entries or scripts.

5) **<u>COM and ActiveX Automation</u>:**

a. **In-Process COM with AutoCAD OEM**: ObjectARX and ObjectDBX applications running in-process with the AutoCAD OEM have full access to the **ActiveX** object model (except for the AutoCAD OEM not supported features);

b. **Out-of-Process COM with AutoCAD OEM**: AutoCAD OEM does not provide direct access to its ActiveX object model from outside its process. To make this scenario work you will need to create a **COM wrapper** to expose those interfaces you will need to access outside AutoCAD OEM process;

c. **AutoCAD OEM inside a Host Application**: AutoCAD OEM can run inside a **child window** of a host application or inside a **Web page** as an **ActiveX control**. In both cases the ActiveX control should be properly configured to work;

d. **VBA**: **DVB** files can be executed inside AutoCAD OEM but they need to be bound (stamped). AutoCAD OEM **does not support embedded VBA macros** in drawings. If a drawing contains embedded macros they will be preserved but not executed;

Creating you AutoCAD OEM product

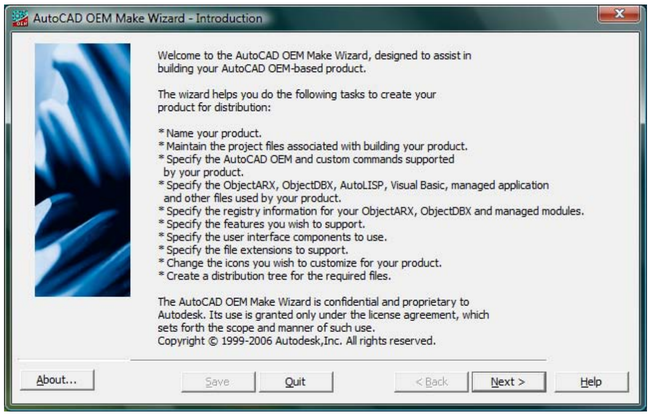
AutoCAD OEM security is based on the **bind (stamp) mechanism**. This mechanism adds stamp information to almost all important modules, to AutoCAD OEM itself and to your custom modules.

This process can be very time-consuming to be done manually. Exactly due that, Autodesk provides a great tool to automate and organize the AutoCAD OEM product build process through logical and simple steps. This tool is called **AutoCAD OEM Make Wizard** which is installed by default.

To start it, open **Windows command prompt**, next:

- Change your current path to "**C:\Program Files\Microsoft Visual Studio 8\VC\bin\**" (may vary depending on your Visual Studio installation);
- Run **vcvars32.bat** command (this will start Visual Studio environment variables);
- Change the path again to "**C:\Program Files\AutoCAD OEM 2008\Toolkit\**";

- Run **OEMMAKEWIZARD.EXE** (Figure 3).



Welcome to the AutoCAD OEM Make Wizard, designed to assist in building your AutoCAD OEM-based product.

The wizard helps you do the following tasks to create your product for distribution:

* Name your product.
* Maintain the project files associated with building your product.
* Specify the AutoCAD OEM and custom commands supported by your product.
* Specify the ObjectARX, ObjectDBX, AutoLISP, Visual Basic, managed application and other files used by your product.
* Specify the registry information for your ObjectARX, ObjectDBX and managed modules.
* Specify the features you wish to support.
* Specify the user interface components to use.
* Specify the file extensions to support.
* Change the icons you wish to customize for your product.
* Create a distribution tree for the required files.

The AutoCAD OEM Make Wizard is confidential and proprietary to Autodesk. Its use is granted only under the license agreement, which sets forth the scope and manner of such use.
Copyright © 1999-2006 Autodesk,Inc. All rights reserved.

Figure 3 – AutoCAD OEM Make Wizard startup screen

A better option is to create a custom **BATCH** file at your desktop with the following commands:

```
C:
CD "C:\Program Files\Microsoft Visual Studio 8\VC\bin\"
CALL vcvars32.bat
CD\
CD "C:\Program Files\AutoCAD OEM 2008\Toolkit\"
OEMMAKEWIZARD.EXE
```

You may also pass command line parameters to the **OEMMAKEWIZARD** application:

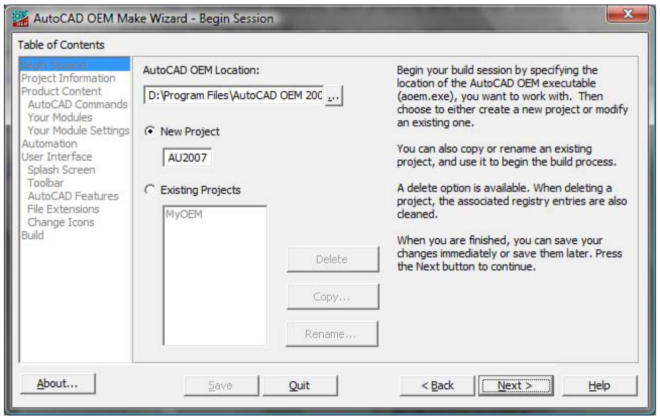
```
<argument1> - Project Name   <argument2> - Keycode

/PA: <path for AutoCAD OEM location>
/PB: <additional paths for bitmap/resources locations>
/BALL - Build All
/BA - Bind your ObjectARX applications
/BV - Bind your DVB macros
/BX - Bind your ObjectDBX applications
/BC - Copy your files
/BL - Bind your AutoLISP applications
/BD - Bind your DLL modules
/BT - Build your type libraries and registry files
/BT+ - Register associated registry files (implies /BT)
/BB – Build your managed applications
/BR – Rebuild your resources
/BR+ - Rebuild your resources and use your settings with AutoCAD OEM
/? – Display command line options
```

The **OEMMAKEWIZARD** organize the process on steps to help you to better setup your branded product through a logical sequence as follows:

STEP 1 – Begin Session

In this step you will manage your **AutoCAD OEM projects**. First, you need to specify where is located the **aoem.exe** module. Under its folder there will be a folder called **Projects** which is where your projects will be created. You may **create**, **delete**, **copy** or **rename** your projects (Figure 4). Once you delete a project its folder and registry settings will be also deleted.

> Click Next to continue



AutoCAD OEM Location: D:\Program Files\AutoCAD OEM 200

[✓ New Project] AU2007

Existing Projects: MyOEM

Begin your build session by specifying the location of the AutoCAD OEM executable (aoem.exe), you want to work with. Then choose to either create a new project or modify an existing one.

You can also copy or rename an existing project, and use it to begin the build process.

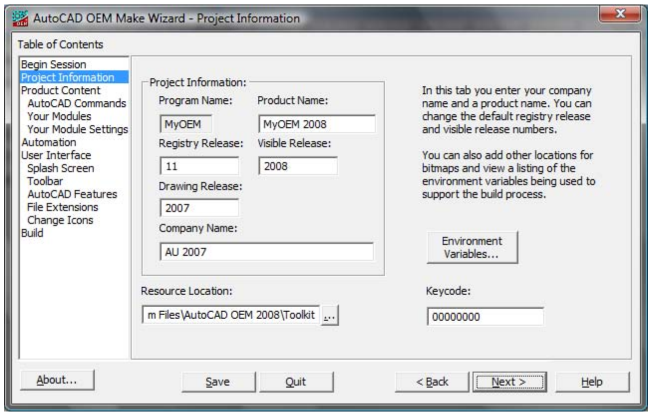
A delete option is available. When deleting a project, the associated registry entries are also cleaned.

When you are finished, you can save your changes immediately or save them later. Press the Next button to continue.

Figure 4 – Begin Session

STEP 2 – Project Information

In this step you will set your **project settings** like name, release versions, etc. (Figure 5). Additionally, you need to specify the **Resource location** (path to search for images and other resources) and the **Keycode** of your product. Optionally, if you want to batch build your product, some **Environment Variables** needs to be specified (click on the button to get them).



Project Information:

Program Name: MyOEM

Product Name: MyOEM 2008

Registry Release: 11

Visible Release: 2008

Drawing Release: 2007

Company Name: AU 2007

Resource Location: m Files\AutoCAD OEM 2008\Toolkit

Keycode: 00000000

In this tab you enter your company name and a product name. You can change the default registry release and visible release numbers.

You can also add other locations for bitmaps and view a listing of the environment variables being used to support the build process.

Figure 5 – Project Information

- **Program Name**: This name is the same you gave to the project during its creation and it will be the main EXE module name;
- **Product Name**: This is the product's main window name (up to 20 characters);
- **Company Name**: Your Company name which your product belongs to;
- **Registry Release**: It is a text (up to 10 characters) to identify your product at Windows Registry. The OEMMakeWizard will prefix it with the "R" character. Your product sub-key will be like: "**HKEY_LOCAL_MACHINE\\SOFTWARE\\COMPANY_NAME\\PRODUCT_NAME\\RREGISTRY_RELEASE**" (note the "R" character);
- **Visible Release**: A text displayed at several labels and at VERSION specific OEM system variable;
- **Drawing Release**: A text with the DWG version supported by your product. AutoCAD 2008 supports

2007 DWG version;

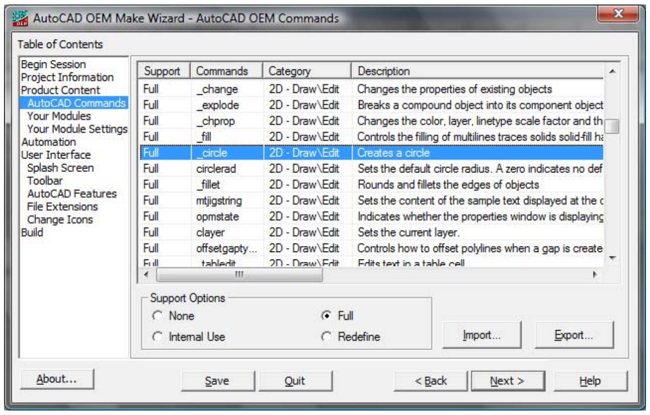
- **Resource Location**: This is the default search path for your resource files that will be used during the build process;
- **Keycode**: The key code need to be formatted as: **MMSSSSSDD**, where **MM** is the current month, **SSSSS** is the last five digits of your AutoCAD OEM serial number, and **DD** is the current day. Save this keycode because it is essential to bind all your product modules. If you plan to release a maintenance package in the future you will need it to be able to build the product patch;
- **Environment Variables**: Display a dialog with all required environment variables required to batch build your product.

> Click Next to continue

STEP 3 – Product Content

STEP 3.1 : AutoCAD Commands

In this step you will specify the availability of **AutoCAD commands.** You need to follow your AutoCAD OEM contract and respect its agreement. This list can be **imported** and **exported** (Figure 6). You may select more than one command to speed up settings. Click at each column header to sort the command list.



Support	Commands	Category	Description
Full	_change	2D - Draw\Edit	Changes the properties of existing objects
Full	_explode	2D - Draw\Edit	Breaks a compound object into its component object
Full	_chprop	2D - Draw\Edit	Changes the color, layer, linetype scale factor and th
Full	_fill	2D - Draw\Edit	Controls the filling of multilines traces solids solid-fill ha
Full	_circle	2D - Draw\Edit	Creates a circle
Full	circlerad	2D - Draw\Edit	Sets the default circle radius. A zero indicates no def
Full	_fillet	2D - Draw\Edit	Rounds and fillets the edges of objects
Full	mtjigstring	2D - Draw\Edit	Sets the content of the sample text displayed at the c
Full	opmstate	2D - Draw\Edit	Indicates whether the properties window is displaying
Full	clayer	2D - Draw\Edit	Sets the current layer.
Full	offsetgapty...	2D - Draw\Edit	Controls how to offset polylines when a gap is create
Full	_tabledit	2D - Draw\Edit	Edits text in a table cell

Support Options: None [✓ Full] Internal Use Redefine

Figure 6 – AutoCAD Commands

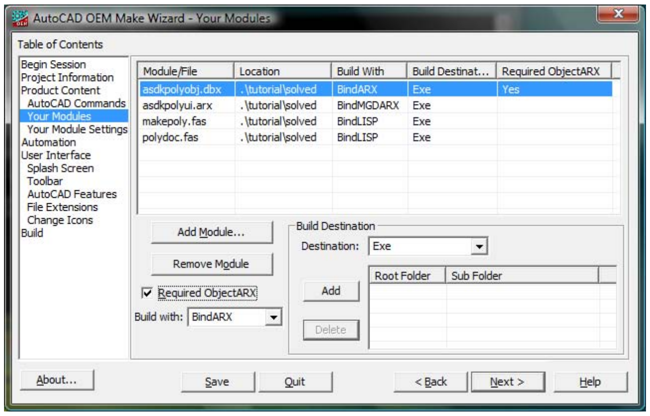
Support Options:

- None: The command will **not** be supported;
- Full: The command will be **fully** supported by your product;
- Internal use: The command can be used only **internally** (it will not be exposed through interfaces);
- Redefine: Your project settings will **redefine** the command using UNDEFINE AutoCAD command;

> Click Next to continue

STEP 3.2: Your Modules

In this step you will specify your **product modules**. These additional modules will add the desired features you want to provide in conjunction with AutoCAD native features. Your modules can be **ARX**, **DBX** (unmanaged or mixed), **DLL** (MFC, C++, Win32 or managed), **FAS** (AutoLISP), **DVB** (VBA) macros and any other custom files you would like to ship with your branded product (Figure 7).



AutoCAD OEM Make Wizard - Your Modules

Module/File	Location	Build With	Build Destinat...	Required ObjectARX
asdkpolyobj.dbx	.\tutorial\solved	BindARX	Exe	Yes
asdkpolyui.arx	.\tutorial\solved	BindMGDARX	Exe	
makepoly.fas	.\tutorial\solved	BindLISP	Exe	
polydoc.fas	.\tutorial\solved	BindLISP	Exe	

[✓] Required ObjectARX

Build with: BindARX

Build Destination — Destination: Exe

Figure 7 – Your Modules

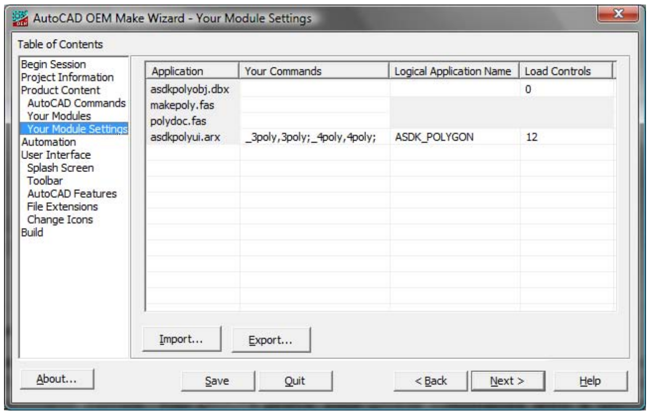
- **Add Module**: Add a new module to the list;
- **Remove Module**: Remove the selected module(s);
- **Required ObjectARX**: Check this option to create a hard dependency between one module and the main EXE module of your product. In practice, if the required module is missing the application will not start (this slow down the startup process but make your product more secure);
- **Build with**: Select the proper action accordingly with the module (CopyFile will just copy the file);
- **Destination**: Select the relative folder where your module will be installed at the target machines. Note that if you need per-user files you must choose "Per-user support". You may also create your own sub-folders with "User Definable" option. "**Add**" and "**Delete**" buttons will manage the custom folders list.

> Click Next to continue

STEP 3.3: Your Module Settings

In this step you will setup each of your modules (Figure 8). Depending on the module type, some options like **commands**, **logical application name** and **load controls** will be allowed to be specified.

At the bottom, there are two buttons to "**Import**" or "**Export**" these settings.



AutoCAD OEM Make Wizard - Your Module Settings

Application	Your Commands	Logical Application Name	Load Controls
asdkpolyobj.dbx			0
makepoly.fas			
polydoc.fas			
asdkpolyui.arx	_3poly,3poly;_4poly,4poly;	ASDK_POLYGON	12

Figure 8 – Your Modules Settings

- **Application**: This column lists the modules you have added at the previous step;
- **Your Commands**: Add here a list of available commands implemented by your modules (if applied). This list consists of a pair of both global (language independent) and local (language specific) command names. You should prefix your global commands with an underscore to make them different from the local versions (**<_global name>**, **<local name>**). If you don't put your registered commands in this list they will not be available through any AutoCAD OEM interfaces like the command prompt, menus and toolbars;

- **Logical Application Name**: The logical application name should match the name that the application uses to registers itself (this is available only for ObjectARX, ObjectDBX and managed applications);
- **Load Controls**: These are options used by AutoCAD OEM to control how your module will be loaded by the main module (this values may be added generating a single number):

  0x01: Load the ObjectARX application when its objects are loaded via drawing open, dxfin, insert, and so on;

  0x02: Load the ObjectARX application when AutoCAD starts up;

  0x04: Load the ObjectARX application whenever a command is executed for which it has a registry entry;

  0x08: Allow loading of the ObjectARX application via the AcRxDynamicLinker::loadApp() method;

  0x10: Do not demand load the ObjectARX application for any reason;

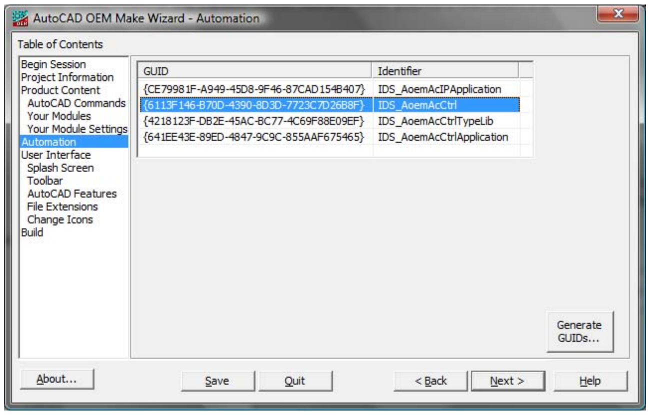
  0x20: Load the ObjectARX application transparently.
- **Import / Export**: Allow you to backup / restore your command settings. Each text file line must have the following format:
  <ModuleName>#[<CommandList>]#[<LogicalApplicationName>]#[<LoadControlsNumber>]

> Click Next to continue

STEP 4 – Automation

In this step you can get the suggested **GUID** identifiers that will be used to allow your product to be used through **Automation** (Figure 9). Your product will act as an **In-Place Server** and this will allow the **AoemAcCtrl** control to be embedded into a web page or a child window. To generate new GUID identifiers, click at the "**Generate GUIDs…**" button.

> Click Next to continue



GUID	Identifier
{CE79981F-A949-45D8-9F46-87CAD154B407}	IDS_AoemAcIPApplication
{6113F146-B70D-4390-8D3D-7723C7D26B8F}	IDS_AoemAcCtrl
{4218123F-DB2E-45AC-BC77-4C69F88E09EF}	IDS_AoemAcCtrlTypeLib
{641EE43E-89ED-4847-9C9C-855AAF675465}	IDS_AoemAcCtrlApplication

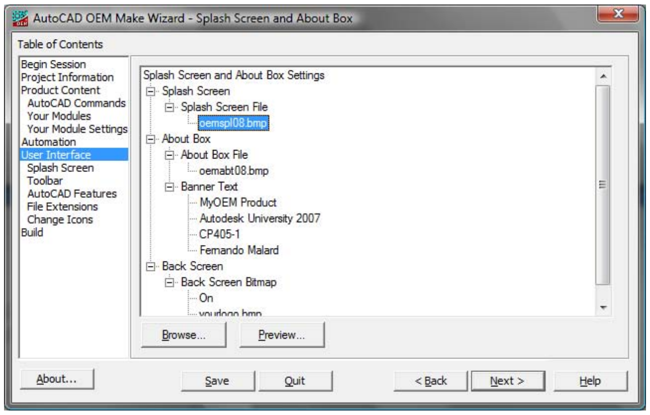
Figure 9 – Automation

STEP 5 – User Interface

STEP 5.1: Splash Screen

You may want, and probably will, change the default **bitmaps** of your AutoCAD OEM product. This can be done through this page by selecting each feature you would like to change (Figure 10). Refer to AutoCAD OEM documentation about the bitmap sizes and color depth. Use the "**Browse**" and "**Preview**" buttons to change and visualize your settings.

> Click Next to continue



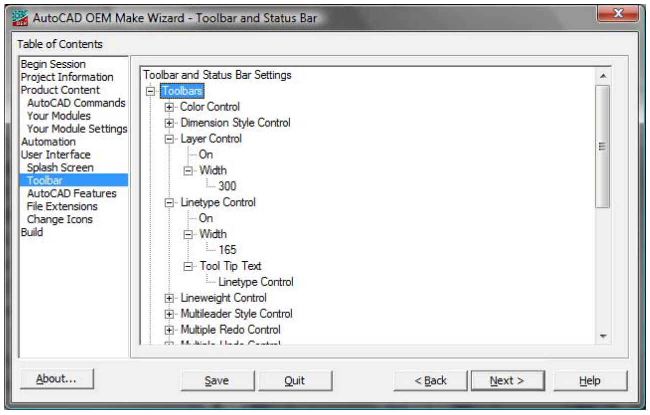
```
Splash Screen and About Box Settings
⊟ Splash Screen
    ⊟ Splash Screen File
        oemspl08.bmp
⊟ About Box
    ⊟ About Box File
        oemabt08.bmp
    ⊟ Banner Text
        MyOEM Product
        Autodesk University 2007
        CP405-1
        Fernando Malard
⊟ Back Screen
    ⊟ Back Screen Bitmap
        On
        yourlogo.bmp
```

Figure 10 – User Interface

STEP 5.2: Toolbar

At this page you can change **Toolbars**, **Dash Boards**, **Status bar** and other interface features (Figure 11). This is a great tool to make your product interface clean and direct displaying only what your end users need to see and use.

> Click Next to continue



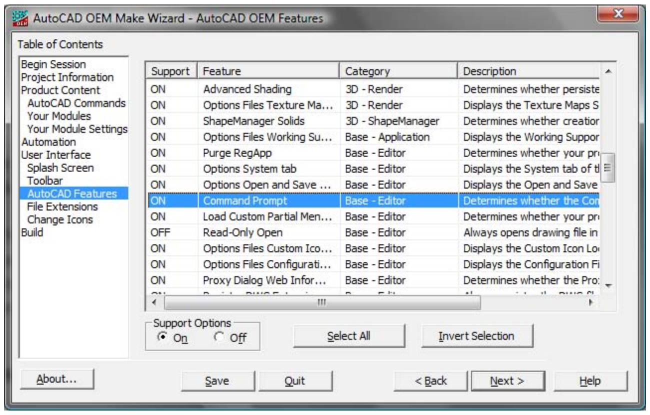
```
Toolbar and Status Bar Settings
⊟ Toolbars
    ⊞ Color Control
    ⊞ Dimension Style Control
    ⊟ Layer Control
        On
        ⊟ Width
            300
    ⊟ Linetype Control
        On
        ⊟ Width
            165
        ⊟ Tool Tip Text
            Linetype Control
    ⊞ Lineweight Control
    ⊞ Multileader Style Control
    ⊞ Multiple Redo Control
```

Figure 11 – Toolbar

STEP 5.3: AutoCAD Features

In this step you will turn **ON**/**OFF** those **AutoCAD features** you will provide with your product (Figure 12). This must be used accordingly with your AutoCAD OEM Contract. Use the "**Select All**" and "**Invert Selection**" buttons to speed up your setup. Click at each column header to sort the list.

> Click Next to continue



Support	Feature	Category	Description
ON	Advanced Shading	3D - Render	Determines whether persiste
ON	Options Files Texture Ma...	3D - Render	Displays the Texture Maps S
ON	ShapeManager Solids	3D - ShapeManager	Determines whether creatior
ON	Options Files Working Su...	Base - Application	Displays the Working Suppor
ON	Purge RegApp	Base - Editor	Determines whether your pr
ON	Options System tab	Base - Editor	Displays the System tab of tl
ON	Options Open and Save ...	Base - Editor	Displays the Open and Save
ON	Command Prompt	Base - Editor	Determines whether the Con
ON	Load Custom Partial Men...	Base - Editor	Determines whether your pr
OFF	Read-Only Open	Base - Editor	Always opens drawing file in
ON	Options Files Custom Ico...	Base - Editor	Displays the Custom Icon Lo
ON	Options Files Configurati...	Base - Editor	Displays the Configuration Fi
ON	Proxy Dialog Web Infor...	Base - Editor	Determines whether the Pro:

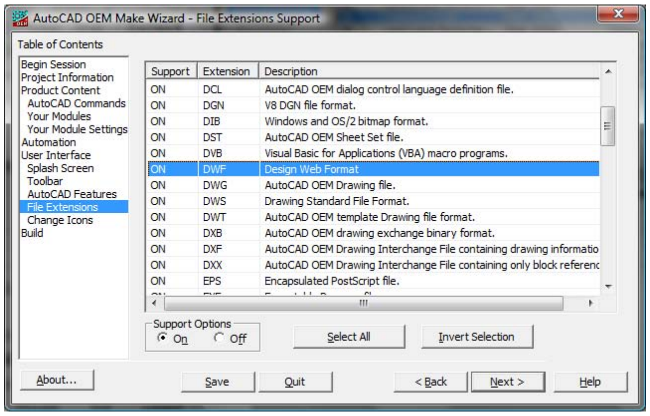
Support Options [✓ On] Off

Figure 12 – AutoCAD Features

STEP 5.4: File Extensions

In this step you will turn **ON**/**OFF** those **AutoCAD file extensions** you will support in your product (Figure 13). Use the "**Select All**" and "**Invert Selection**" buttons to speed up your setup. Click at each column header to sort the list.

> Click Next to continue



Support	Extension	Description
ON	DCL	AutoCAD OEM dialog control language definition file.
ON	DGN	V8 DGN file format.
ON	DIB	Windows and OS/2 bitmap format.
ON	DST	AutoCAD OEM Sheet Set file.
ON	DVB	Visual Basic for Applications (VBA) macro programs.
ON	DWF	Design Web Format
ON	DWG	AutoCAD OEM Drawing file.
ON	DWS	Drawing Standard File Format.
ON	DWT	AutoCAD OEM template Drawing file format.
ON	DXB	AutoCAD OEM drawing exchange binary format.
ON	DXF	AutoCAD OEM Drawing Interchange File containing drawing informatio
ON	DXX	AutoCAD OEM Drawing Interchange File containing only block referenc
ON	EPS	Encapsulated PostScript file.

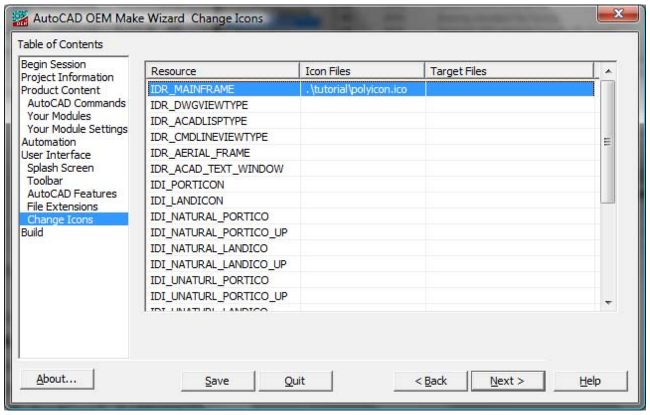
Support Options: [✓ On] Off

Figure 13 – File Extensions

STEP 5.5: Change Icons

In this step you will customize your product icons (Figure 14). At the **Icon Files** column, you can choose the file containing the new icon image. At the **Target Files** column, you can click on the "**Browse**" button to specify the file which the icon will be changed to, including **<program name>.exe** and **<program name>res.dll**.

> Click Next to continue



Resource	Icon Files	Target Files
IDR_MAINFRAME	.\tutorial\polyicon.ico	
IDR_DWGVIEWTYPE		
IDR_ACADLISPTYPE		
IDR_CMDLINEVIEWTYPE		
IDR_AERIAL_FRAME		
IDR_ACAD_TEXT_WINDOW		
IDI_PORTICON		
IDI_LANDICON		
IDI_NATURAL_PORTICO		
IDI_NATURAL_PORTICO_UP		
IDI_NATURAL_LANDICO		
IDI_NATURAL_LANDICO_UP		
IDI_UNATURL_PORTICO		
IDI_UNATURL_PORTICO_UP		

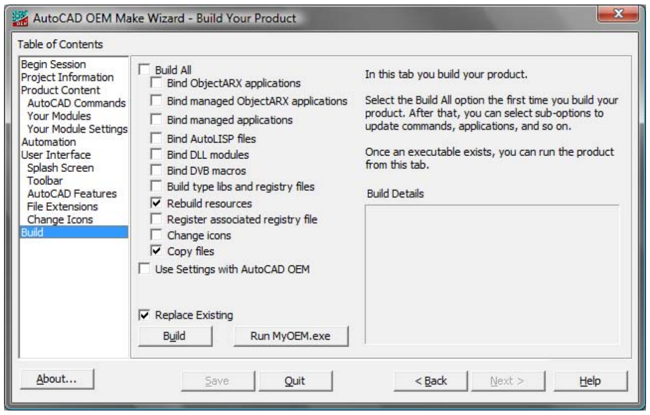
Figure 14 – Change Icons

STEP 6 – BUILD

In this step you will finally build your product with all AutoCAD chosen features plus your own modules (Figure 15). The first build need to be done with the "**Build All**" option enabled.

This process will take some time because every single module of AutoCAD plus yours will need to be **bound to the main EXE module**.

This security procedure will guarantee that those applications will only run into your product and will not run outside it.



- [ ] Build All
  - [ ] Bind ObjectARX applications
  - [ ] Bind managed ObjectARX applications
  - [ ] Bind managed applications
  - [ ] Bind AutoLISP files
  - [ ] Bind DLL modules
  - [ ] Bind DVB macros
  - [ ] Build type libs and registry files
  - [✓] Rebuild resources
  - [ ] Register associated registry file
  - [ ] Change icons
  - [✓] Copy files
- [ ] Use Settings with AutoCAD OEM
- [✓] Replace Existing

In this tab you build your product.

Select the Build All option the first time you build your product. After that, you can select sub-options to update commands, applications, and so on.

Once an executable exists, you can run the product from this tab.

Figure 15 – Build

- **Build All**: This option perform a FULL build and need to be done at the first time;
- **Bind ObjectARX applications**: Binds your ObjectARX/ObjectDBX modules;

- **Bind managed ObjectARX applications**: Binds only your mixed-mode modules;
- **Bind managed applications**: Binds only your managed applications (.NET modules);
- **Bind AutoLISP files**: Binds only your compiled AutoLISP files (FAS files);
- **Bind DLL modules**: Binds only the DLL modules (MFC / Win32 files);
- **Bind DVB macros**: Binds the VBA macros (DVB files);
- **Bind type libraries and registry files**: Binds only automation modules and registry files;
- **Rebuild Resources**: Build your resource files like bitmaps;
- **Register Associated Registry file**: Add your product registry entries to Windows Registry;
- **Change icons**: Apply icons your have specified at Change Icons page;
- **Copy files**: Copy your modules with CopyFile option (these files will not be bound);
- **Use Settings with AutoCAD OEM**: Rebuilds aoemres2.dll and scree.dll using your product's settings. Copies the original aoemres2.dll and scree.dll files into the AutoCAD OEM directory as aoemres2.org and scree.org (backup files). Useful to test your modules inside default AutoCAD OEM;
- **Replace Existing**: Instruct the Wizard to overwrite existing files during build process;

The "**Build**" button will start the building process. This process is essentially a batch sequence that will perform several tasks. This process may take a while to finish and a command line window will be opened eventually during this process. The time consumed is proportional to the above options you have enabled. If you are only change some of these files you don't need to enable other options.

The "**Run**" button will start your built product.

> Click Quit to exit (always remember to save your project)

Creating and Testing ObjectARX Modules

ObjectARX applications need to be fully compiled and built with specific AutoCAD OEM SDK. Essentially, this SDK is almost the same of native AutoCAD ObjectARX SDK but it enables your modules to be loaded at your OEM branded product and only inside it.

After installing your AutoCAD OEM there will be specific **include** and **library** files which you will need to point to from inside your **Visual Studio**.

AutoCAD OEM is installed with its own sample version which has all AutoCAD features enabled to provide you a perfect environment for developing and testing your product modules.

After building your ObjectARX applications, using the AutoCAD OEM include and library files, you need to patch the application modules so that they can be loaded into **aoem.exe** (the AutoCAD OEM application). The **patcher.exe** program, provided at the C:\Program Files\AutoCAD OEM 2008\toolkit directory, will do this for you. The syntax to apply the patch is as follows:

```
patcher -IZ <application name> "AOEMAOEM.EXE\0" "AOEM.EXE\0" [-warn]
```

In other hand, there is a much better alternative to test your product at a real environment. Using your Visual Studio, perform the following steps:

- Build you ARX/DBX modules using the **OEM ObjectARX SDK** (include and library files);
- Add your compiled ARX/DBX modules in the "**Your Modules**" page;

- Add any commands defined within the “**Your Module Settings**” page;
- Build the OEM product from the “**Build**” page;
- Inside **Visual Studio** settings of your main ARX project, set **<PROD>.exe** that is posted in the \Program Files\AutoCAD OEM 2008\oem\<prod> folder as the executable to run;
- Use **F5** to start the debug process;
- Next time, perform changes to your code and **rebuild only your ARX/DBX modules**;
- Each time after building, go back to the “**Build**” page, selecting only the “**Bind ObjectARX modules**” and “**Rebuild my Resources**” check boxes, along with “**Replace Existing**”, and build the OEM application;
- Go back to Visual Studio and click **F5** again to debug…

The second approach will allow you to fully test your product with only its final features and thus allowing you to detect possible problems.

Creating and Testing AutoLISP Modules

Inside **aoem.exe**, AutoCAD OEM provides the **Visual LISP® IDE**, which you can use to develop and test your AutoLISP applications. You will need to use Visual LISP IDE to run your AutoLISP application because your commands are not registered at the default AutoCAD OEM product. This way, you need to call your commands using the following syntax:

(C:YOUR_COMMAND)

At your branded product you will be able to use **FAS** / **FSL** (AutoLISP compiled files) modules that can be loaded without any problem. End users cannot enter AutoLISP expressions using the **keyboard**, **menu**, or **script** files inside your branded product. Additionally, stamped AutoCAD OEM products cannot access the **debug-enabled**, **run-time expression evaluator**.

Unfortunately, there are some other severe limitations. Specifically, although bound **FAS** files are supported in AutoCAD OEM, you cannot create and use **VLX** files (Visual LISP packed applications). Further, you cannot use Visual LISP functions related to **ActiveX®** interfaces, including the **vl-load-com**, **vlax-*** and **vlr-*** functions.

To **compile** your AutoLISP applications, say a **mymodule.lsp** module, do the following:

- Start AutoCAD OEM and the **Visual LISP® IDE** using the **VLISP** command.
- Enter: (vlisp-compile 'st "C:/Your_Project_Source/mymodule.lsp")
- This command produces a compiled **mymodule.fas** file which can now be added to your project at “**Your Modules**” page. It will be stamped and thus allowed to run inside your branded product.

Creating and Debugging VBA Modules

AutoCAD OEM provides the **VBA IDE** in the **aoem.exe** program. You can use the VBA IDE to develop and test your **VBA macros** directly in AutoCAD OEM. On the other hand, you cannot use VBA IDE inside your branded products. VBA is shipped with AutoCAD OEM and is available to developers. Only **DVB** files that are bound can be loaded and run in AutoCAD OEM products.AutoCAD OEM VBA support is provided in the ObjectARX application **acvba.arx**. You must ship acvba.arx with your product if it provides any VBA commands. You must also ensure that VBA is loaded by your product. If you do not supply acvba.arx, VBA will be disabled.

The following commands are available in both **unstamped** and **stamped** applications:

- VBALOAD, -VBALOAD, VBARUN, -VBARUN, VBAUNLOAD, VBAMAN (Load and Unload options).

The following commands are available for use in **aoem.exe** and disabled in stamped products:

- VBAIDE, VBANEW, VBASTMT, VBAMAN (New, Extract, SaveAs and IDE options only), VBAPREF (Allow Break on Errors option only).

The **VBAMAN** (Embed option) command is disabled in both unstamped and stamped applications. AutoCAD OEM does not support embedded VBA macros in drawings. Embedded macros are preserved in DWG filing operations, but they are not executed.

Creating an Installation Package

Once you have your branded AutoCAD OEM product tested and ready to ship you can use the **OEMInstallerWizard** application to generate your installation package. AutoCAD OEM implements Microsoft **WindowsInstaller** (**MSI**) technology to comply with the most current Windows application standards.

Before packing your product you may want to change or keep AutoCAD default icon resources. The **aoemficn.dll** file is installed by the AutoCAD OEM and contains default icons. If you want only to keep the default icons you just need to copy this file and rename it to **<program name>ficn.dll**. Next, add it to "**Your Modules**" page with **CopyFile** option. You can replace existing icons in **aoemficn.dll**, but you cannot remove icons from this DLL. If you add icons, their file names must begin with a "ZZZ" prefix to assure that they are added to the end of the existing icon list. If you want to change these icons and/or add some new icons to this DLL, you will need to recompile this DLL using **Visual Studio** to build a new version of it. This DLL's Visual Studio project is located at: **\InstallerSample\Aoemficn** into your AutoCAD OEM CD (more about this process at **oemdev.chm** help file).

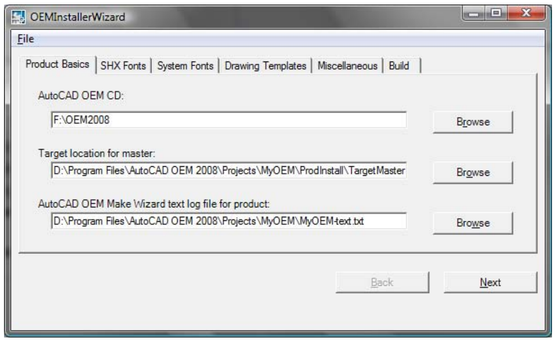
To start the **OEMInstallerWizard**:

- **Copy** the **InstallerWizard** directory from the AutoCAD OEM CD to your computer;
- In the InstallerWizard directory, double-click **oeminstallerwizard.exe**;

Product Basics

In this page you need to provide where is your **AutoCAD OEM CD**, your desired **Target** folder and where is your **OEMMakeWizard** log file (Figure 16). Use the "**Browse**" buttons to specify these fields.

> Click Next to continue



AutoCAD OEM CD: F:\OEM2008

Target location for master: D:\Program Files\AutoCAD OEM 2008\Projects\MyOEM\ProdInstall\TargetMaster

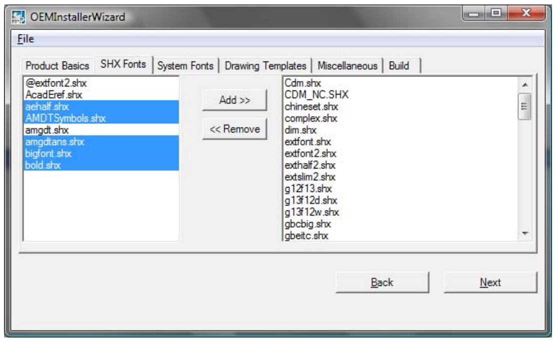
AutoCAD OEM Make Wizard text log file for product: D:\Program Files\AutoCAD OEM 2008\Projects\MyOEM\MyOEM-text.txt

Figure 16 – Product Basics

SHX Fonts

In this page you will select those SHX fonts you would like to ship with your product (Figure 17). Use the "**Add>>**" and "**<<Remove**" buttons to manage these fonts.

> Click Next to continue



@extfont2.shx
AcadEref.shx
aehalf.shx
AMDTSymbols.shx
amgdt.shx
amgdtans.shx
bigfont.shx
bold.shx

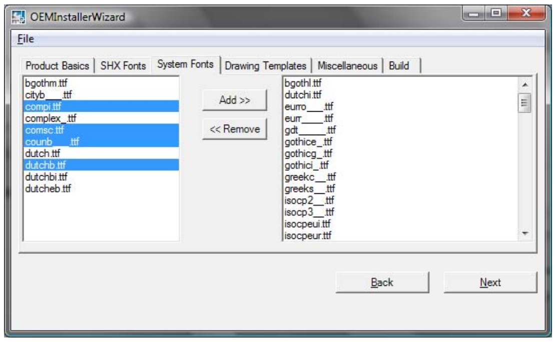
Cdm.shx
CDM_NC.SHX
chineset.shx
complex.shx
dim.shx
extfont.shx
extfont2.shx
exthalf2.shx
extslim2.shx
g12f13.shx
g13f12d.shx
g13f12w.shx
gbcbig.shx
gbeitc.shx

Figure 17 – SHX Fonts

System Fonts

In this page you will select the System fonts (TTF) you would like to ship with your branded product (Figure 18). Use the "**Add>>**" and "**<<Remove**" buttons to manage these fonts.

> Click Next to continue



bgothm.ttf
cityb___.ttf
compi.ttf
complex_.ttf
comsc.ttf
counb___.ttf
dutch.ttf
dutchb.ttf
dutchbi.ttf
dutcheb.ttf

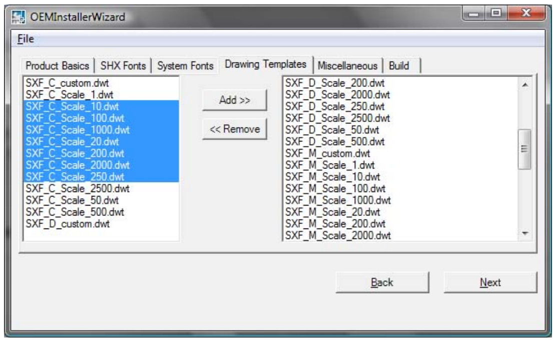
bgothl.ttf
dutchi.ttf
eurro___.ttf
eurr____.ttf
gdt_____.ttf
gothice_.ttf
gothicg_.ttf
gothici_.ttf
greekc__.ttf
greeks__.ttf
isocp2__.ttf
isocp3__.ttf
isocpeui.ttf
isocpeur.ttf

Figure 18 – System Fonts

Drawing Templates

In this page you will select the **DWT** (drawing template) files to ship with your product (Figure 19). Use the "**Add>>**" and "**<<Remove**" buttons to manage these template files.

> Click Next to continue



SXF_C_custom.dwt
SXF_C_Scale_1.dwt
SXF_C_Scale_10.dwt
SXF_C_Scale_100.dwt
SXF_C_Scale_1000.dwt
SXF_C_Scale_20.dwt
SXF_C_Scale_200.dwt
SXF_C_Scale_2000.dwt
SXF_C_Scale_250.dwt
SXF_C_Scale_2500.dwt
SXF_C_Scale_50.dwt
SXF_C_Scale_500.dwt
SXF_D_custom.dwt

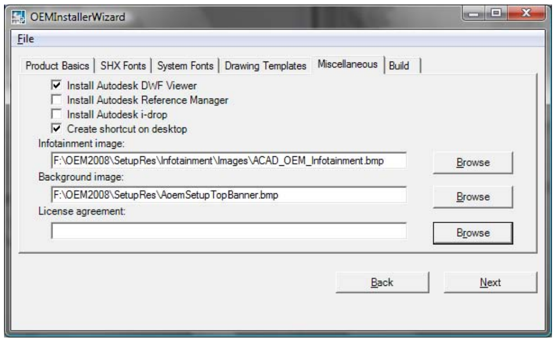
SXF_D_Scale_200.dwt
SXF_D_Scale_2000.dwt
SXF_D_Scale_250.dwt
SXF_D_Scale_2500.dwt
SXF_D_Scale_50.dwt
SXF_D_Scale_500.dwt
SXF_M_custom.dwt
SXF_M_Scale_1.dwt
SXF_M_Scale_10.dwt
SXF_M_Scale_100.dwt
SXF_M_Scale_1000.dwt
SXF_M_Scale_20.dwt
SXF_M_Scale_200.dwt
SXF_M_Scale_2000.dwt

Figure 19 – Drawing Templates

Miscellaneous

In this page you will specify **additional features** installed with your branded product (Figure 20). Use the "**Browse**" buttons to select desired files.

> Click Next to continue



☑ Install Autodesk DWF Viewer
☐ Install Autodesk Reference Manager
☐ Install Autodesk i-drop
☑ Create shortcut on desktop

Infotainment image: F:\OEM2008\SetupRes\Infotainment\Images\ACAD_OEM_Infotainment.bmp

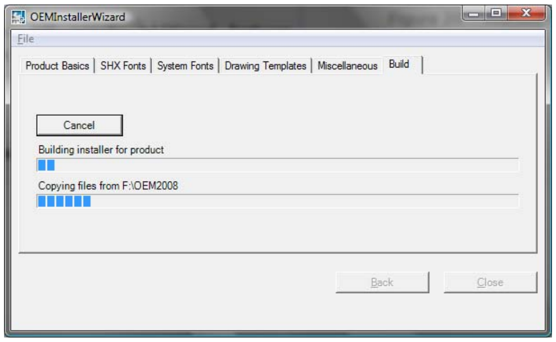
Background image: F:\OEM2008\SetupRes\AoemSetupTopBanner.bmp

License agreement:

Figure 20 – Miscellaneous

Build

Finally, click on the "**Build**" button and wait for the Wizard to complete the **installation package** creation process (Figure 21). If your installation project was successfully finished you can now test the resulting application from inside your **Target** folder. Use the "**Close**" button to exit the Wizard. Remember to save your installation project using the **File** menu.



Building installer for product

Copying files from F:\OEM2008

Figure 21 – Build

> Click Close to finish

You can use **Windows Installer** tools to merge modules into your **MSI** file. For example, using a <u>free utility</u> named **Orca** that is provided with the **Windows Installer SDK**, you can merge modules into your MSI file and edit it.

MSI files created with **InstallShield™** and **Wise™ for Windows Installer** can invoke a nested MSI file created with the OEMInstallerWizard. MSI files created with the OEMInstallerWizard have been tested to run successfully when nested in other MSI packages.

Creating an Installation Patch (Unsupported)

Your AutoCAD OEM branded product will certainly add several **new features** and this will generate **eventual bugs**. Your development team will need to regularly **release bug fixes** through application patches. Your product users will need an easy way to update their CAD stations.

Officially, AutoCAD OEM does not provide an **automatic** or **manual** tool to perform this task. From my own experience, within my development team, there is an **unsupported** way to do that which has been proved to work well at our client install base. The idea behind this solution is to deploy **only the necessary files** (including all your application modules). This will generate **small packages** that can be easily installed on the target machines with a minimum effort.

The first thing to do is to **fully rebuild** your AutoCAD OEM product with the updated version of your modules. Before you do that, you need to **set back the date** of your computer to the same date you have released the first version. In fact you will need to use this same date to all of your future patches. Once you adjust the date and after fully rebuilding your product you can create a small install package containing only your changed modules plus an essential file called **Scree.dll**. This file is responsible for the **stamp security** and it is sensitive to each build you make. Exactly due that you need to ship it every time you create a new build.

<u>Note</u>: *This procedure has worked since AutoCAD OEM 2006 and through 2007 and 2008. It is not guaranteed that Autodesk won't change anything that makes this procedure to stop working. It is recommended that you fully test this procedure before deploying the product to your client machines.*

Applying AutoCAD Service Packs to your OEM Product

Autodesk release at least one **Service Pack** after each release of AutoCAD. This is happening every year since AutoCAD R14 and will probably continue to happen. You will also need to handle these updates on your AutoCAD OEM branded product.

Unfortunately, due the **stamp protection**, you cannot apply the same Service Pack Autodesk make available to all AutoCAD users. First the normal Service Pack is not made to OEM products. Additionally, the files inside the Service Pack are not bound to same EXE module your OEM branded product has. The product main EXE module will refuse to load these Service Pack files.

Today, your only option is to follow an update procedure documented at **Autodesk Buzzsaw** website to **ISV Partners** (your company will need to be member of this community). From this site you will get a specific Service Pack version made exclusively for AutoCAD OEM branded products plus an update tool to apply it to your AutoCAD OEM files.

The bad news here is that you will need to **fully deploy** your application. Your clients will need to first uninstall their current version to be able to install the new version containing the Service Pack. This sounds very annoying but it is the only way to do that by now.

Conclusion

In this session you have learned how to create your own AutoCAD branded product using AutoCAD OEM technology. There is much more information about AutoCAD OEM technology inside ADN (Autodesk Developer Network) and through Autodesk ISV channel.

All of its session material will be made available through **AU-Online** website at http://www.autodesk.com/auonline right after this session. You may also contact me by e-mail to solve your issues related to this course or related to AutoCAD OEM. Further I would like to invite you to visit my **Blog** site at http://arxdummies.blogspot.com/ to find out more information about AutoCAD OEM and ObjectARX technologies.

Thank you!