



V 6.0

# Table of Contents

## **Chapter 1. Introduction to the IGES/Works Users' Guide 6**

*Who should read this manual? 6*

*How is this book organized? 6*

*IGES/Works Script Sharing 7*

## **Chapter 2. Introduction to IGES/Works 8**

*Ways to use IGES/Works 8*

Modifying a File to Meet Desired Design Standards 8

Inspecting a File 8

Extracting the Data You Need 8

"Curing" a "sick" File 9

## **Chapter 3. Learning the Basics 12**

### **About Using IGES/Works 12**

*Starting IGES/Works 12*

*Working With the windows 15*

Starting IGES/Works 15

Opening a File 15

Working with Entities 18

*Working with the Dialog Boxes 19*

*A Typical IGES/Works Session 20*

Working with a File 20

Selecting Entities 20

Working With Selected Entities 20

Saving and Closing the File 21

*Quitting IGES/Works 21*

## **Chapter 4. Working with files 22**

*Opening, Saving, and Closing Files 22*

Opening a File 22

Saving a File 23

Closing a File 23

*Configuring IGES/Works Defaults 23*

Configuring IGES/Works from the Options Menus 23

Editing the Configuration File 28

*Manipulating the File View 35*

About the Display Manipulation Controls 35

*Printing a file 36*

How do you print a file? 36

*Validating a file 36*

What is validation? 36

Why validate a file? 37

How do you Validate a File? 37  
What do the Validation and Fix-up Messages Look Like? 39  
How do you Interpret the Validation Messages? 42

*Projecting a File 43*

What is Projection? 43  
Why Project a File? 43  
How do you Project a File? 43

*Working with the Start Section of a File 43*

Listing the Start Section 44  
Copying the Start Section 44  
Extracting Data from the Start Section 44  
Modifying Data in the Start Section 44

*Working with the Global Section of a File 44*

Listing the Global Section 44  
Copying the Global Section 44  
Extracting Data from the Global Section 44  
Modifying the Global Section 45

**Chapter 5. Working with Entities 46**

*Selecting entities 46*

What is the Current Selection List? 46  
Ways to Select Entities 46  
Manipulating selection lists from the command line 48

*Getting more Information About Entities 49*

*Editing Entities 50*

**Chapter 6. Working with Conversions 54**

*Converting entities 54*

Why convert entities? 54  
How do you convert entities? 54  
How do you fix Trimmed Surface entities? 54  
How do you reverse surface normals? 54

**Chapter 7. Running Scripts 58**

*What are Scripts? 58*

*Why use Scripts? 58*

*Running Scripts 59*

Running Scripts from the Scripts Menus 59  
Running Scripts from the Custom Menus 60  
Running Scripts from Inside Another Script 60  
Running Scripts from the Command Line 60

**Chapter 8. Entering IGES/Works Commands 61**

*Syntax of IGES/Works Commands 61*

Form 1 61  
Form 2 62  
Form 3 62  
Form 4 62

*Using Aliases for IGES/Works Commands* 63  
*Command Line Features* 63  
*Working with Message Output* 64  
*Working with Operating System Commands* 65  
*Using and Modifying Wild Card Characters* 65

## **Chapter 9. Understanding the Data Definition File 67**

*The Data Definition File Representation of the IGES Specification* 67  
*Referencing an Entity's PD Data* 68  
*Referencing an Entity's DE Data* 71  
*Referencing an Entity's Property Pointers* 73  
*Referencing an Entity's Associativity Pointers* 73  
*Referencing an Entity's Derived Fields* 74

## **Chapter 10. Navigating the IGES Entity Hierarchy 75**

*Using the Change Selection Command* 75  
*Using the Find Entity Command* 77

## **Chapter 11. Working with IGES/Works Variables 80**

*About IGES/Works variables* 80  
*Creating IGES/Works variables* 80  
*About IGES/Works System Variables* 81  
*Using IGES/Works Variables in Commands* 82  
*About IGES/Works Functions with Variables* 82  
    Math Functions 83  
    String Manipulation and Conversion Functions 84  
    Matrix Functions 84  
    Other Functions 85  
*Using IGES/Works Functions with Variables* 86

## **Chapter 12. Writing Scripts 87**

*Writing and Modifying Scripts* 87  
    Command Journaling 87  
    GUI recording 88  
    Switches of the Execute Script Command for Debugging 88  
    Passing Data into the Script 88  
    Using Conditional Statements 92  
    Using Loop Structures 94  
    Issuing Messages from the Script 95

## **Chapter 13. Working with Generic Files 96**

*About using ASCII Files* 96

*Generating Reports from ASCII Files* 96

*Reading ASCII Files* 96

Sample ASCII File Reading Script 97

Sample input ASCII Datafile 'lines.dat' 101

**Appendix A. Introduction to IGES 102**

*What is IGES?* 102

*Why you Should use IGES* 103

**Appendix B. Scripts 105**

**Appendix C. Tolerances 110**

**Glossary 112**

## Chapter 1. Introduction to the IGES/Works Users' Guide

Welcome to IGES/Works, the software that allows you to analyze, display, edit, and modify IGES files.

### ***Who should read this manual?***

This manual is intended for anyone needing to troubleshoot, or modify IGES files to meet the design conventions of a customer or supplier. Whether you are creating an IGES translation test case, or just need to extract pertinent data from a file, you can use this manual to help you find the right IGES/Works tools for the job.

### ***How is this book organized?***

<b>Chapter</b>	<b>Description</b>
<b>Chapter 2.</b> Introduction to IGES/Works	Explains how IGES/Works can help enhance your data exchange environment
<b>Chapter 3.</b> Learning the Basics	Explains how to enter and exit IGES/Works, and how to use its interface.
<b>Chapter 4.</b> Working with Files	Describes the basics of working with files, including manipulating your view of the IGES data, and validating files.
<b>Chapter 5.</b> Working with Entities	Describes the tools IGES/Works provides for extracting the data you need, or modifying the content, attributes, and structure of a file.
<b>Chapter 6.</b> Working with Conversions	Explains how to convert entities to other IGES equivalents and perform cleanups on curves for trimmed surfaces.
<b>Chapter 7.</b> Running Scripts	Explains what scripts are, what you can use them for, and how to run existing scripts.
<b>Chapter 8.</b> Entering IGES/Works Commands	Describes the IGES/Works command syntax. It also explains how to use wild card characters and aliases.
<b>Chapter 9.</b> Understanding the Data Definition File	Describes the syntax of IGES entity definitions in the Data Definition File.
<b>Chapter 10.</b> Navigating the IGES Entity Hierarchy	Explains how to use IGES/Works' Change Selection and Find Entity commands to navigate the entity hierarchy structure of an IGES file.
<b>Chapter 11.</b> Working with IGES/Works Variables	Explains how to create and manipulate variables to store values when creating scripts and using command modifiers
<b>Chapter 12.</b> Writing Scripts	Describes the basics of writing scripts to create tailor-made functions. It covers passing data into and out of scripts, using conditional statements and loop structures, and issuing messages.
<b>Chapter 13.</b> Working with Generic Files	Explains how to use IGES/Works to generate reports by accessing data in non-IGES, ASCII files. It also explains how you can create IGES entities from raw data contained in ASCII files.
<b>Appendix A,</b> "Introduction to IGES"	Describes the IGES neutral data exchange format; its pros and cons.
<b>Appendix B,</b> "Scripts"	Describes the ready-to-use scripts shipped with IGES/Works.

<b>Appendix C</b> , "Tolerances"	Describes how to solve tolerance problems when you are working with the display and performing conversions{XE "Conversion: solving tolerance problems"} and validations.
<b>Appendix D</b> , "License Manager"	Describes XNET License Manger commands for querying and configuring
<b>Glossary</b>	Explains terms used in this manual and other IGES/Works documentation that are specific to IGES/Works.

### ***IGES/Works Script Sharing***

A directory is set up to contain user shared scripts. This directory contains many of the scripts that were available on the script sharing Bulletin Board at the time of release.

Custom IGES/Works scripts can be downloaded or uploaded to ITI.

To Access Scripts via Internet:

Type:

```
ftp itimail.iti-oh.com
```

Log in as "anonymous"

Use your e-mail address as your password

(You are now on the system)

Type:

```
cd /pub/igesworks
```

## Chapter 2. Introduction to IGES/Works

IGES/Works is the tool kit that allows you to analyze, display, edit, and modify IGES files. You can use it to

- verify the IGES file before sending and upon receiving
- analyze and pinpoint sources of data exchange problems
- develop solutions for improving IGES data exchange
- customize the IGES file to the capabilities of the receiving system
- convert between differing design conventions of the sending and receiving companies
- standardize the data exchange environment and make it easier for general usage by using customizable dialogs and menus.

Whether you are creating an IGES translation test case, or just need to extract pertinent data from a file, IGES/Works has the right tools for the job.

### ***Ways to use IGES/Works***

The way you use IGES/Works depends on what you need to get from the file or on the types of problems it presents. Your needs probably fall into one or more of the following categories:

- modifying a file to meet desired design standards
- inspecting a file
- extracting the data you need
- “curing” a “sick” file

### **Modifying a File to Meet Desired Design Standards**

In many cases you will want to use IGES/Works' editing and standards conformance tools to make IGES files adhere to your company's design standards or those of your customers and suppliers.

### **Inspecting a File**

Sometimes all you need is to look at a file to make a visual determination about what is wrong with it. IGES/Works provides a variety of tools for manipulating the visual display of IGES files. You may also want to use IGES/Works to print a log of all the invalid entities it contains.

### **Extracting the Data You Need**

It may be that your goal is not to save the entire file, but rather to extract only those pieces of it that you need to work with. You will want to use IGES/Works' entity selection{XE "Selecting entities:extracting data"} tools in combination with its editing tools. For example, there may be some specific structures you need from a file. You can open the file in IGES/Works, select the structures you want, and extract them to a separate file. Conversely, you could select and delete{XE "Deleting entities"} all the entities you do not want from the original file.

## “Curing” a “sick” File

In some cases, you may want to use IGES/Works' validation tool to automatically diagnose and fix an IGES file. In other cases you may want to use IGES/Works to help you make your own diagnosis and design your own solution.

IGES/Works comes with a number of tools that allow you to do this.

The table below shows the relationships between IGES file “diseases,” their “symptoms,” and IGES/Works “cures.” In some cases the suggested cure will be enough. In other cases, you will need to write scripts. Scripts allow you to customize the cure.

These symptoms	Mean these diseases	Use these cures.
<p><b>Main:</b> A translator crashes during file processing, or the CAD system crashes while activating the newly created part.</p> <p><b>Secondary:</b> The part comes up but large chunks of data are missing.</p> <p><b>Secondary:</b> The part comes up, but some of the data is mutated or misplaced.</p>	<p>Bugs! The IGES file contains a case (valid or not) that it cannot handle.</p> <p>Entity (or entities) is not supported.</p>	<p>Validate the file.</p> <p>Develop scripts.</p> <p>Convert the entity to a supported one.</p>
<p><b>Main:</b> The part comes up, but some of the data is mutated or misplaced.</p>	<p>The IGES specification was implemented incorrectly.</p>	<p>Validate the file.</p> <p>Develop scripts.</p>
These symptoms	Mean these diseases	Use these cures.
<p><b>Main:</b> The part comes up but large chunks of data are missing.</p> <p><b>Secondary:</b> The part displays well, but the data does not have the correct structure.</p>	<p>A translator was not updated to the latest version of the CAD system or IGES.</p>	<p>Simplify the structures.</p> <p>Insert new data.</p> <p>Develop scripts.</p>

<b>Main:</b> The part displays well, but the data does not have the correct structure.	The entities and entity structures used in the source system differ from those in the target system or in IGES.	Simplify the structures. Insert new data.
<b>Secondary:</b> The part comes up but large chunks of data are missing.		

### *Validating the File*

It is often difficult, if not impossible to navigate through the ASCII IGES file to look for errors. IGES/Works allows you to automatically validate all entities against a set of constraints (their IGES definitions as well as their logical meaning, i.e., zero length line).

When IGES/Works discovers an error, it makes syntax and semantic changes to a file to make it comply with the IGES specification. For example, validation may force the start and end points of an arc to be equi-distant from its center.

### *Simplifying Structures*

IGES files often contain structures too complex for the target system. IGES/Works provides the following tools for simplifying file structures:

- Projection
- Decomposition
- Conversion

**Projection** simplifies an IGES file by flattening its 3D entities. IGES/Works projects the file through some view into its 2D equivalent.

**Decomposition** explodes structure entities by making the children independent and deleting the structure. In the case of subfigures, decomposition simplifies an IGES file by creating actual entities for every defined entity instance. For example, an IGES file might contain a nut (consisting of six lines and an arc) defined once as a subfigure and then instanced one hundred times. The file would contain six lines, one arc, one subfigure definition, and one hundred instances. Decomposing the file creates the properly oriented six hundred lines and one hundred arcs.

**Conversion** simplifies an IGES file by replacing unsupported entities with supported entities as a tool for simplifying file structures. For example, IGES/Works can change all parametric splines in the file into NURBS curves.

You can combine these techniques according to the type of structures you are working with to make any IGES file as simple as you need it to be. For example you could go so far as to turn an entire file into line segments.

The following table provides guidance on the choice of simplification tool for specific types of structure.

<b>For these structures</b>	<b>Simplify this way</b>
Drawing/View	Projection
3D entities	Projection
Subfigures	Decomposition{XE "Decomposition:structure simplification tool"}
Groups	Decomposition via scripts
Dimensions	Decomposition
Curves and Surfaces	Conversion
Trimmed Surfaces	Conversion
Bounded Surfaces	Conversion
Solids	Decomposition

### *Inserting New Data*

Adding structures or entities to an IGES file may be a necessity (in the case of a “sick” file), or just a convenience. For example, you might need to add a drawing to a 3D model, put all entities on a certain level into a subfigure, or add name properties to views.

## Chapter 3. Learning the Basics

*In this chapter you learn about*

- using IGES/Works
- starting IGES/Works
- dynamic display
- working with the windows
- working with the dialog boxes
- a typical IGES/Works session
- quitting IGES/Works

### About Using IGES/Works

This chapter describes the fundamentals of starting and using IGES/Works. Use this chapter to develop a general sense of orientation. Read the chapters on working with files, entities, and scripts for a more detailed description of specific aspects of using and configuring IGES/Works. Consult the on-line help for guidance on using specific window and dialog box features. If you need information about using the command line, refer to the *IGES/Works Command Reference Manual*.

#### Note

*IGES/Works includes the Motif graphical user interface (GUI) on Unix machines, but uses the standard interface on Windows NT. While the functionality of the software is the same in both cases, some minor details on how you use the interface may differ slightly. For specific information on using your particular interface consult your interface documentation.*

#### Tip

*IGES/Works also allows you to enter commands through a command line. Most buttons and menu items, plus commands entered at the command line, are journaled as IGES/Works commands. Refer to the GUI recorder portion of Chapter 12, for details.*

You may also want to consider using the command line and configure IGES/Works for non-graphic mode (see the table on the next page) when graphics is not available. Advanced users may want to use commands not available from the GUI. Refer to the *IGES/Works Command Reference Manual* for details.

You can add commands not available via the GUI to the Custom menus{XE "Custom menu:commands not available via the GUI:journaling"} – commands you execute in this manner are journaled.

### Starting IGES/Works

Invoke IGES/Works by entering the name of the executable and zero or more options:

```
igesworks [IGES file name] [-c configuration file {XE "Configuration file:starting IGES/Works"}{XE "Starting IGES/Works"}name] [-display machinename] [-ng] [-j] [-e script file name [args]]
```

#### For NT users:

To use the startup features, you may create a new icon or change an existing one. These startup options may also be used from the dos prompt. When specifying files remember to include two / marks.

Where	Means
[ IGES file name]	Specifies the name of an IGES file to be processed. If this argument is provided, IGES/Works will begin with the file read in, made current, validated and displayed (or whatever the open file options indicate should be done).
NT users:	
[ //IGES file name]	
[ -c configuration file name ]	Specifies the name of a configuration file{XE "Configuration file:starting IGES/Works:specifying name of configuration file"}. Default is <code>./igesworks</code> or <code>~/igesworks</code> or nothing (the configuration file is not required).
NT users:	Note: NT configuration file is <code>iw.cfg</code>
[ /c configuration file name ]	
[-display machinename]	Specifies the name of the machine to display IGES/Works. If not specified, the value of the DISPLAY environment variable is used. By default the display will go to UNIX, the console device.
[ -ng ]	Brings up IGES/Works in non-graphics mode. Only command-line directives can be given and no graphics-oriented directives are accepted.
Not supported for NT users	
[-j ]	Brings up IGES/Works in the journaling mode, which means to run all the commands from the journal (file name indicated in the configuration file{XE "Configuration file:running commands from the journal"}) except the last one, which will be only echoed to the screen.
NT users:	
[/j ]	
[ -e script file name ]	Specifies the name of a script file to be executed. This takes the place of the old batch file, with the difference being that IGES/Works does not automatically quit when the script is completed.
NT users:	
[ /e script file name ]	
[ args ]	Data values that are passed into the script. Arguments specified here will be stored in IGES/Works global variables <code>argv[0]</code> , <code>argv[1]</code> , etc. with the number of arguments stored in the global variable <code>argc</code> .



## Examples:

Using a different config file:

```
igesworks -c demo/.igesworks
```

Starting up with an IGES file directly read in:

```
igesworks myfile.igs
```

Executing a batch script:

```
igesworks -ng -e flavor.scr option2
```

## Note to Lite users:

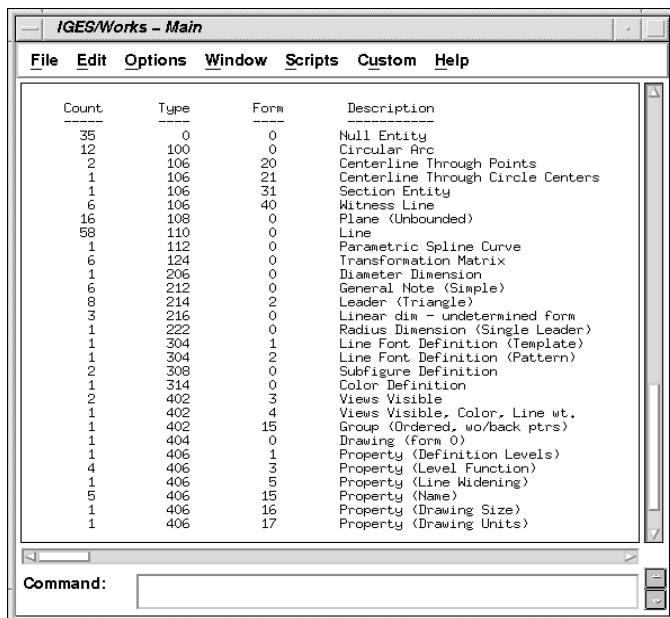
*When starting IGES/Works, remember to include the -l option for Unix platforms and the V option for the NT.*

## Working With the Windows

In using IGES/Works you will encounter three different types of windows: The Main window, the File window, and the Selection List window.

## Starting IGES/Works

The Main window appears.



Use the Main window's pull-down menus to open a file (File:open), configure a variety of IGES/Works options, and run scripts. You can also use this window's command line to enter any IGES/Works command. Text output from commands appears in the Main window's text area.

## Working with the Dynamic Display

There are three functions that will become quite useful:

1. Zoom - zooms the geometry in or out, depending on the mouse movement.

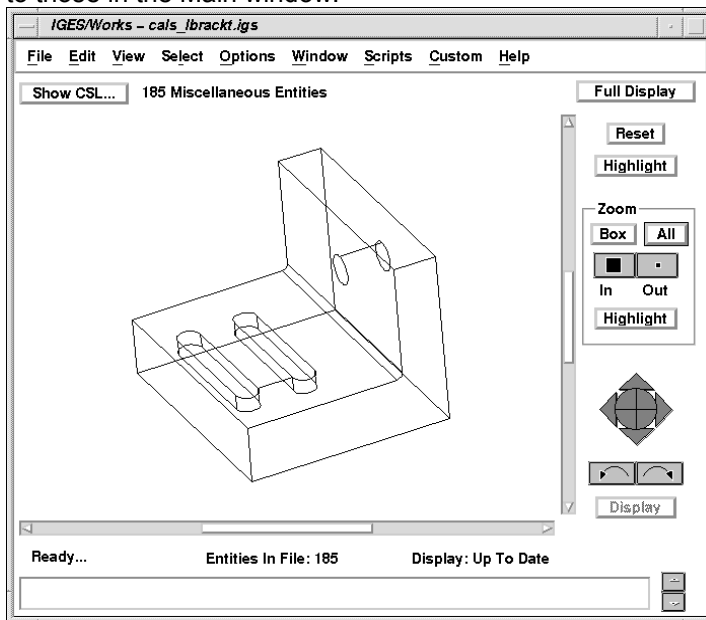
2. Pan - moves the geometry in the direction of mouse cursor and captures the last position of cursor.
3. Rotate - rotates the geometry by the angle calculated. If the geometry is in “shading mode”, it temporarily changes to “Wireframe mode” until the mouse button is pressed. It re-enters the “shading mode” when the mouse button is released.

These buttons change fonts to indicate the currently selected display mode. Click the same button to exit this display mode. The buttons act as toggle switch. Clicking the same button turns it on/off. If you click another button, the display mode may change as indicated by the changed fonts and the mouse cursor.

## Opening a File



It appears in a File window, which has its own set of pulldown menus – some of which are similar to those in the Main window.



The File window allows you to work with the file's entities. Like the Main window, it also has a command line.

### Note:

*Text output from commands appears in the Main window{XE "Main window:using the text display area"}.*

The File window display area shows a graphical representation of the file. Status fields show (top, next to the Show CSL button) a summary of the Current Selection List content and (bottom, above the command line) IGES/Works working status, entity count, and display status.

This window provides a number of tools you can use to manipulate the graphical display and to work with individual and groups of entities. See “When you work with entities ...” below, for additional information about selecting entities.

IGES/Works allows you to have more than one File window open at a time – so you can copy entities{XE "Copying entities:from one file to another"} from one file to another, or compare two files.

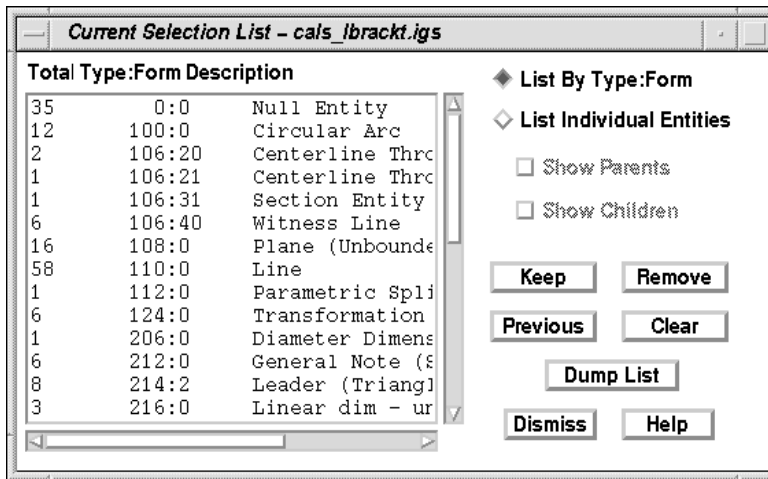
### Note:

*The maximum number of files you can have open at any one time is fifteen. This maximum number is dependent on the amount of temporary disk space available on your system. The File window in which you place the cursor becomes the active window.*

## Working with Entities

Show CSL...

You must first select them{XE "Selecting entities:before you can work with them"} and place them in a selection list. IGES/Works commands work on either the whole file or on entities in the Current Selection List. See Chapter 4, "Working with Files," and Chapter 5, "Working with Entities" for more information about this.



IGES/Works offers a number of ways you can select entities{XE "Selecting entities:variety of methods"} and get detailed information about them. If you have a visual display of the file in the File window, you can click on the entities you want. You can also select{XE "Selecting entities:a particular class"} a particular class of entities by using either the File window Select pulldown menu or the buttons on the Current Selection List window.

### Tip

*While you are working with the Current Selection List, you can save it and come back to it later. You can even create a variety of different selection lists for the file.*

Once you have all the entities you want in the Current Selection List, you can work with them. For example, you can

- edit them (delete{XE "Deleting entities:working with the Current Selection List"}, copy{XE "Copying entities:working with the Current Selection List"}, modify{XE "Modifying entities:working with the Current Selection List"}, etc.),
- convert them,
- run a script designed to perform a certain action on entities it finds in the Current Selection List.

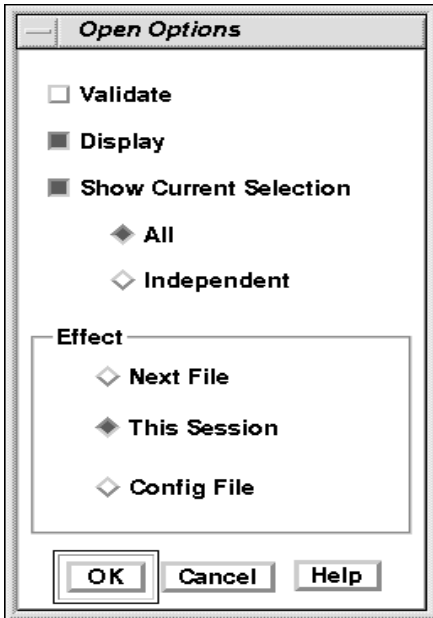
### Note:

*Only one selection list can be the Current Selection List at any one time.*

For more information on selecting entities and working with the selection lists, see "Selecting entities" in Chapter 5, "Working with entities."

## **Working with the Dialog Boxes**

When you execute some IGES/Works commands, IGES/Works displays a dialog box to prompt you for more information about how to complete the command.



Depending on which dialog {XE "Dialog:working with"}box you are working with, you may be required to

- select one or more items from lists
- choose buttons – for mutually exclusive options
- select check boxes where multiple options can be applied
- and click control buttons to either execute the command or cancel it.

For example, the Open Options dialog {XE "Dialog:Open Options"} box prompts you to

- select what actions you want to be automatically performed when you open an IGES file
- choose when you want IGES/Works to apply the actions you selected.

You can get general information and specific instructions for working with a dialog {XE "Dialog:working with:getting general information and specific instructions "} box by consulting its online help.

## ***A Typical IGES/Works Session***

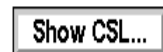
How you work with IGES/Works depends on the nature of the problem you are using it to solve, and on how you configure the various IGES/Works options. Whatever the case, your IGES/Works session will usually follow this same pattern.

### **Working with a File**



Start from the Main window by opening a file {XE "File:open"}. In selecting the file you want, you will work with a standard File Open dialog {XE "Dialog:File Open"} box. The file appears in its own File window. You can either examine the file for yourself or you can have IGES/Works automatically validate {XE "Validation:as an automatic File Open option"} it. You can also choose from a number of other tools that allow you to work with the file as a whole. You may also want to work with individual or groups of entities.

### **Selecting Entities**



If you need to dig deeper, use any of the several techniques available to select the entities {XE "Selecting entities:using the File window and Selection List window"} you want to work with. You will find you are constantly moving back and forth between the File window and the Selection List window. You may be selecting entities by clicking on them in the graphical display, or by using the File window's Select menu and the Selection List window buttons. When the list is complete, you are ready to work with the selected entities.

### **Working With Selected Entities**



The File window's Edit menu allows you to edit {XE "Editing entities>Edit menu options"} the selected entities. You can

- copy {XE "Copying entities:from one file to another"} them and paste them into a new file
- delete {XE "Deleting entities:working with the Current Selection List"} them
- modify them by changing their attributes, DE (Directory Entry) or PD (Parameter Data) values
- convert them into a different entity type
- run a script on them.

## Saving and Closing the File



When you are done working with the file and its entities, use the File window's File menu to save{XE "File:save"}, then close the file{XE "File:close"}.

### **Quitting IGES/Works**

When you are done working with IGES/Works, use either the Main or the File window's File menu to Exit.

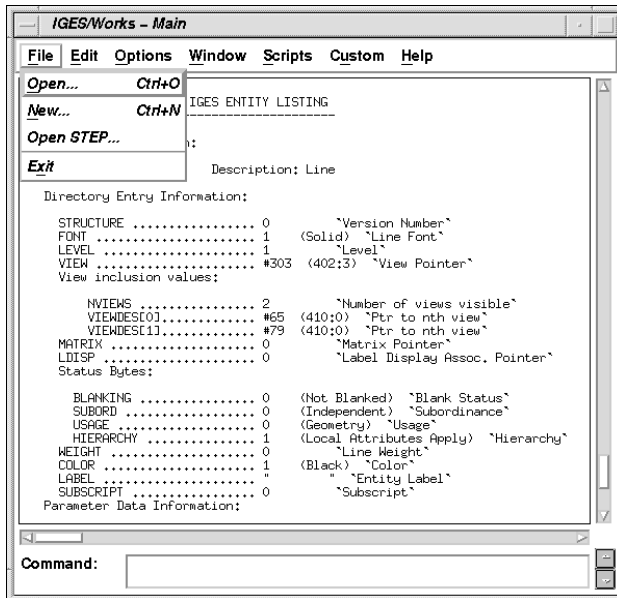
## Chapter 4. Working with files

In this chapter you learn about

- opening, saving, and closing files
- configuring IGES/Works defaults
- manipulating the file view
- validating a file
- projecting a file
- working with the Start section of a file
- working with the Global section of a file

### Opening, Saving, and Closing Files

Whether you are opening, saving, or closing a file, you can do all this from the File window File menu. You can also open files from the Main window File menu.



### Opening a File

When you first start IGES/Works you can open a file from the File menu in the Main window. Once you have a File window open, you can open other files from either the Main window or the File window File menu. Which window you use to do this will depend on convenience – which window you are currently working in.

Whether you open a file from the Main window or the File window, IGES/Works displays a standard file open dialog {XE "Dialog:File Open"}, allowing you to choose the file you want to work with.

You can also open a new file – one that contains no data – into which you can copy data from other files or create new entities{XE "Creating entities: in a new file"}. In which case, IGES/Works does not present you with a file open dialog, instead it directly presents an empty File window. If you are opening an IGES file and it is taking too long, you can interrupt it by pressing the “Q” key. This will stop processing. If you do this, you may not be able to use some commands or update the display since the model may not be in a valid state. If you experience unpredictable behavior after interrupting the processing, you should close the model.

## Saving a File

You can save a file{XE "File:save"} from the File menu in its own File window. The first time you save a file IGES/Works prompts you for a filename with a standard file save dialog{XE "Dialog:File Save"}.

## Closing a File

You can close a file{XE "File:close"} from the File menu of its own File window. If you have not already saved the file, IGES/Works prompts you to choose whether to close it without saving it or not.

## Configuring IGES/Works Defaults

At startup time IGES/Works refers to a configuration file to find out how it should configure its various options for the session. There are three main ways IGES/Works finds this configuration file{XE "Configuration file:specifying which one IGES/Works uses"}:

- you can specify a configuration file in the command line options when you invoke IGES/Works
- if you don't specify a configuration file, IGES/Works looks first in the local directory, and then in your home directory
- if IGES/Works fails to find a configuration file in any of the above places, it uses the internal defaults which are similar to those in the default configuration file (named .igesworks) in the bin directory.

There are two ways you can change IGES/Works configuration defaults:

- from the Main and File window Options menus
- by editing the configuration file.

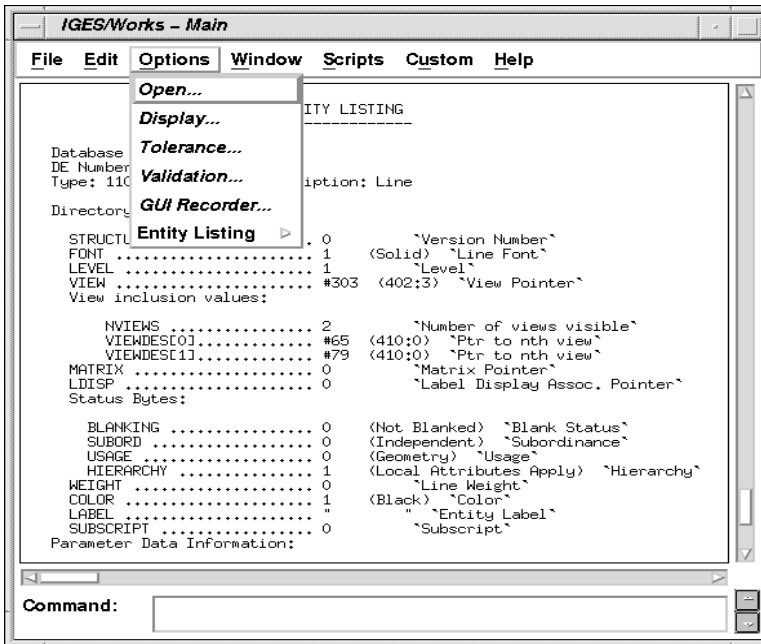
See “Starting IGES/Works” in Chapter 3, “Learning the basics” for details on specifying a configuration file name.

## Configuring IGES/Works from the Options Menus

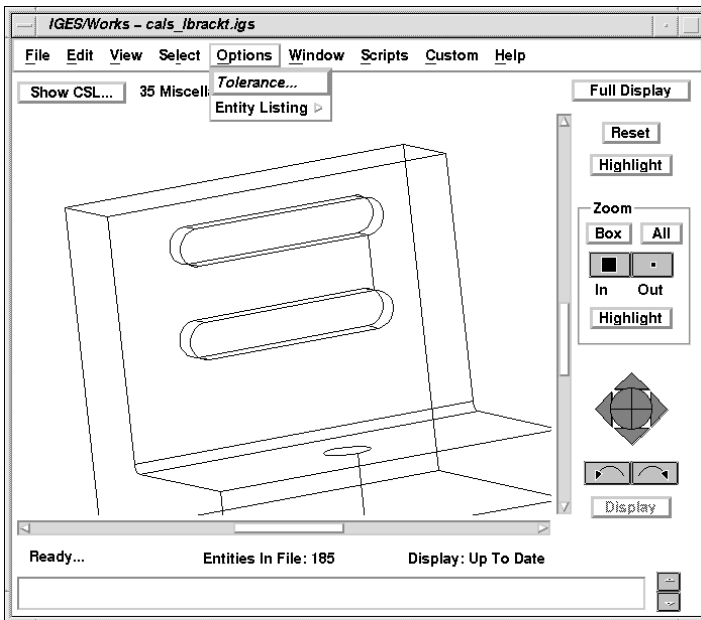
Once IGES/Works is up and running you can configure its Open, Display, Entity listing, and Tolerances{XE "Tolerances:configuring options"} options from the Main window Options menu.

**Use the Main window Options menu** if you want to choose the range over which these configuration options take effect:

- the next file you open
- all files you open in this session
- write them to the configuration file IGES/Works started with – making them the default options next time you start IGES/Works with that configuration file{XE "Configuration file:changing default options"}. These options are also valid for the remainder of this session.



**Use the File window Options menu** to configure tolerance values you want to apply only to the file you are currently working with. Display and Validation options (Validation:setting options) are set from the File Menu on the File Window, when you want to set them for an already opened file.

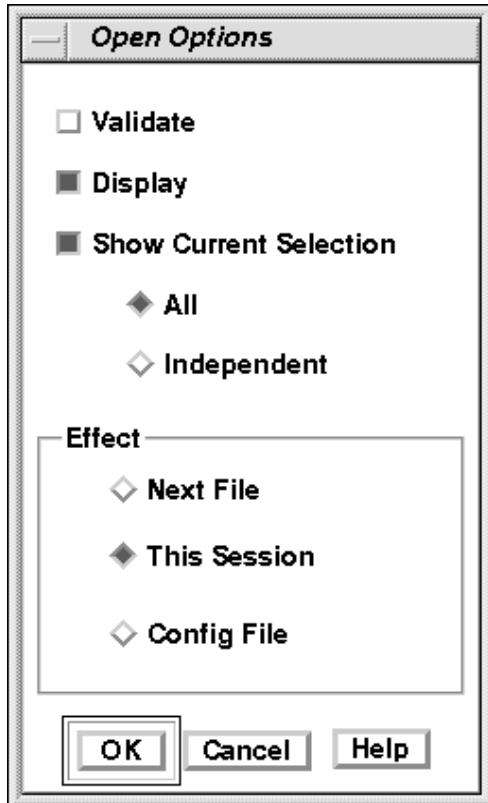


**Note:**

*The File window Options menu does not offer a file Open Options selection, since you already have a file open. But you could go back to the Main window to alter the Open Options from its Options menu if you want to affect what happens when you open subsequent files.*

**File Open Options** allow you to choose what activities you want to occur, automatically, when you open a file. IGES/Works offers the following file Open options:

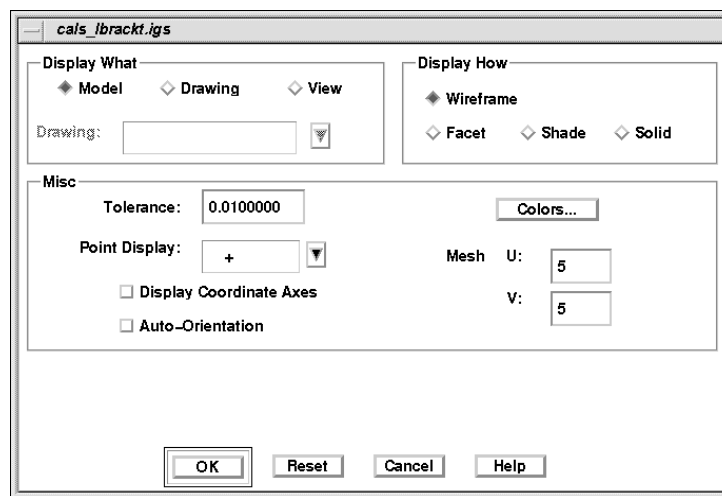
- validate the file
- display the file
- show the Current Selection List



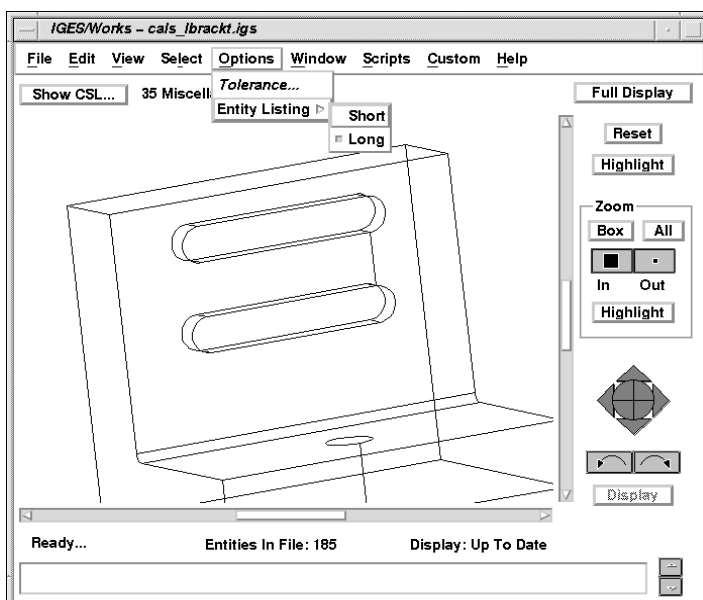
For example, if you choose Display as the file Open option for the next file, IGES/Works will automatically display the next file you open according to the display options you choose.

**Display Options** allow you to configure the way IGES/Works displays the file. IGES/Works offers the following display options:

- what you want to display (model, view, drawing)
- how to display it (wireframe, faceted, shaded, solid)
- how the coordinate axes appear (dashed or solid, and color{XE "Color:display options"})
- auto orientation (enable/disable)
- how points appear (dots/crosses)
- display tolerance{XE "Tolerances:display"} (a one-line value, specifies number of segments used for approximation. For more information on this, refer to the *display model* command in the *IGES/Works Command Reference Manual*).



**Entity Listing Options** allow you to specify whether you want the default entity listing to be the short or the long version.



**Note:**

*You can also switch between the long and the short version when you display the entity listing. This option just allows you to set the default, it does not preclude you being able to display one or the other.*

**Tolerance options allows you to** set the tolerance {XE "Tolerances:setting the default values"} default values by means of a dialog{XE "Dialog:tolerance options"} box.

The screenshot shows a dialog box titled "Tolerances" with a standard Windows-style title bar. The dialog is organized into two columns of settings. Each setting consists of a label and a text input field containing a numerical value in scientific notation. At the bottom of the dialog are four buttons: "OK", "Reset", "Cancel", and "Help".

Setting	Value
Zero	1.0000e-13
Co-planar	1.0000e-08
Epsilon	1.0000e-08
Angle	1.0000e-07
Colinear	1.0000e-07
Parameter Space Point	1.0000e-08
Normal Magnitud	1.0000e-06
Model Space Point	1.0000e-03

These values determine the accuracy of convergence between the original source file entities and the entities that result from validation fixes{XE "Validation:tolerance values"}, entity conversions, and entity approximations{XE "Subset generation:tolerances"}. See Appendix C, "Tolerances" for more information.

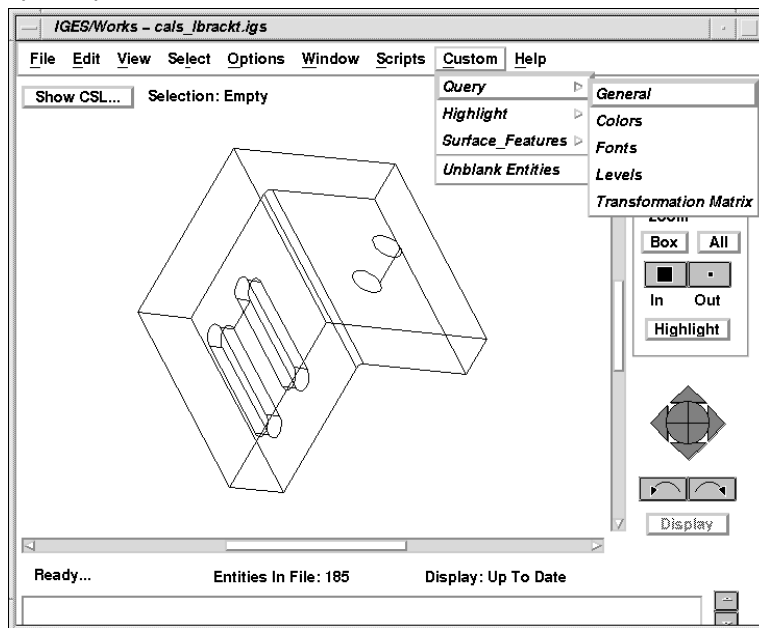
## Editing the Configuration File

You can modify IGES/Works' configuration defaults by using a text editor to edit your configuration files{XE "Configuration file:using a text editor to modify defaults"}.

When you install IGES/Works, a default configuration file (named .igesworks) is placed in the bin directory. You can copy this file to your home directory and edit it to suit your needs.

The default configuration file contains comments. Comments are contained between `"/*` and `"*/`.

- create Custom menus{XE "Custom menu:creating by editing the configuration file"}
- set iges\_dir
- set script\_dir
- set entity\_listing
- set Open Options



The following is a sample configuration file:

```
/* -----  
  
        VARIABLE INITIALIZATIONS  
  
*/  
  
    history_count = 20;  
    screen_size = 0;  
    prompt_var = "";  
    journal_file = "igesworks.jrn";  
    line_width = 90;  
    max_message = 5;  
  
    help_dir = "<PATH>/help";  
    iges_dir = "<PATH>/igesfiles";  
    map_dir = "<PATH>/data";  
    script_dir = "<PATH>/scripts";  
    font_dir = "<PATH>/fonts";  
  
    data_ext = ".dat";  
    iges_ext = ".igs";  
    message_ext = ".msg";  
    report_ext = ".rpt";  
    script_ext = ".scr";  
    smf_ext = ".smf";  
    smt_ext = ".smt";  
  
/*  
-----  
  
        ALIAS INITIALIZATIONS  
  
*/  
  
    ALIAS am = "allocate model name";  
    ALIAS bs = "build script";  
    ALIAS cs = "change selection field_spec";
```

```
ALIAS ch = "clear highlight";
ALIAS ce = "copy entity model";
ALIAS cgs = "copy global_section model";
ALIAS cn = "convert to_nurbs -summary";
ALIAS cl = "convert to_112_114 -summary";
ALIAS css = "copy start_section model";
ALIAS ct = "convert 143to144 -summary";
ALIAS da = "define alias name";
ALIAS dv = "define variable name";
ALIAS dm = "display model";
ALIAS ed = "echo data args";
ALIAS ef = "evaluate function args";
ALIAS es = "execute script file";
ALIAS fe = "find entity selection_spec";
ALIAS fd = "free display";
ALIAS fm = "free model name";
ALIAS fs = "free selection name";
ALIAS fv = "free variable name";
ALIAS gd = "get data args";
ALIAS gf = "get field field_spec";
ALIAS ggf = "get global_field field_spec";
ALIAS gsf = "get start_field field_spec";
ALIAS gm = "generate model_curve";
ALIAS gp = "generate param_curve";
ALIAS hm = "help me";
ALIAS hs = "highlight selection";
ALIAS ip = "invoke process arg";
ALIAS la = "list alias";
ALIAS lc = "list config";
ALIAS ld = "list db_entity";
ALIAS lddf = "list ddf";
ALIAS lgs = "list global_section";
ALIAS lh = "list history";
```

```
ALIAS li = "list IGES_Entity";
ALIAS lmo = "list message_output";
ALIAS lm = "list model";
ALIAS lp = "list parent_data";
ALIAS lscr = "list script";
ALIAS ls = "list selection";
ALIAS lssl = "list ssl";
ALIAS lss = "list start_section";
ALIAS lt = "list type_form";
ALIAS lv = "list variable";
ALIAS mc = "modify config parameter";
ALIAS mf = "modify field field_name";
ALIAS mgs = "modify global_section field_name";
ALIAS mss = "modify start_section field_name";
ALIAS pd = "plot display file";
ALIAS pm = "project model model";
ALIAS qs = "quit system";
ALIAS ri = "read iges file";
ALIAS rmsg = "redirect messages";
ALIAS rm = "restore model file";
ALIAS rs = "restore selection name";
ALIAS svm = "save model name";
ALIAS ss = "save selection name";
ALIAS se = "select entity";
ALIAS sm = "select model name";
ALIAS sd = "set display";
ALIAS vm = "validate model";
ALIAS wi = "write iges file";
ALIAS int = "define variable name";
ALIAS flt = "define variable -float name";
ALIAS fstr = "define variable -fstring name";
ALIAS bye = "quit system -force";
ALIAS logout = "quit system -force";
```

```

ALIAS exit = "quit system -force";
ALIAS fall = "find entity field_spec / -recursive";
ALIAS logfile = "redirect messages -both -all -command -force file";
ALIAS errfile = "redirect messages -file -error -command -force file ";
ALIAS err_prompt = "modify config parameter prompt_var value error_return";
ALIAS csl_prompt = "modify config parameter prompt_var value csl_count";
ALIAS mod_prompt = "modify config parameter prompt_var value cur_model";
ALIAS nest_prompt = "modify config parameter prompt_var value cur_nest";
ALIAS path_prompt = "modify config parameter prompt_var value cur_wp";

/*
-----
ENVIRONMENT VARIABLE INITIALIZATIONS
*/

These environment variables can either be set in the user's
login shell or in here in the IGES/Works configuration file.

Don't forget to remove the comment symbols around the environment
variable that you wish to set!

/*
/*
X DISPLAY variable

/*
/*
ENV DISPLAY = "unix:0.0";

/*
/*

The following environment variable overrides the UNIX temp-file
directory specified by the environment variable TMPDIR.  If neither,
are set, temporary files will be placed in /usr/tmp or /tmp.

/*

/*
ENV PDELIB_TMP = "/usr/tmp";

/*

```

```

/*
    For MOTIF users only.  This environment variable sets
    the path to the igesworks.uid file
*/

/*
ENV UIDPATH = "<PATH>/bin/igesworks.uid";
*/

/*
-----
CUSTOM MENUS
*/

MENU MAIN_CUSTOM_MENU {
    menu Logfile;
};

MENU FILE_CUSTOM_MENU {
    menu Query;
    menu Highlight;
    menu Surface_Features;
    separator;
    "Unblank Entities" "execute script file Unblank_Ents.scr -no_stop";
};

MENU Logfile {
    "Log File on" "redirect messages -both -all -command file igesworks.log -
force";
    "Log File off" "redirect messages";
}

MENU Query {

```

```

"General" "execute script file Query_Dialog.scr -no_stop";
"Colors" "execute script file Query_Colors.scr -no_stop";
"Fonts" "execute script file Query_Fonts.scr -no_stop";
"Levels" "execute script file Query_Levels.scr -no_stop";
"Transformation Matrix" "execute script file Query_TMs.scr -no_stop";
};
MENU Highlight {
    "Invalid Entities" "execute script file Select_Invalid.scr -no_stop";
    "Corrected Entities" "execute script file Select_Fixed.scr -no_stop";
}
Menu Surface_Features {
    "Turn Features On" "show features";
    "Turn Features Off" "show features -no_start_point -no_uv_orig -no_control_pts -no_control_mesh -no_surf_normal -no_curve_direct";
    "Show Control Pts" "show features -no_start_point -no_uv_orig -control_pts -control_mesh -no_surf_normal -no_curve_direct";
    "Show Direction" "show features -no_start_point -uv_orig -no_control_pts -no_control_mesh -surf_normal -curve_direct";
}

```

### Tip

*You can create (and keep for future use) more than one configuration file{XE "Configuration file:invoking IGES/Works with different configuration files"}, each for a different type of session and/or user. Invoke IGES/Works with the desired configuration file by using the -c option.*

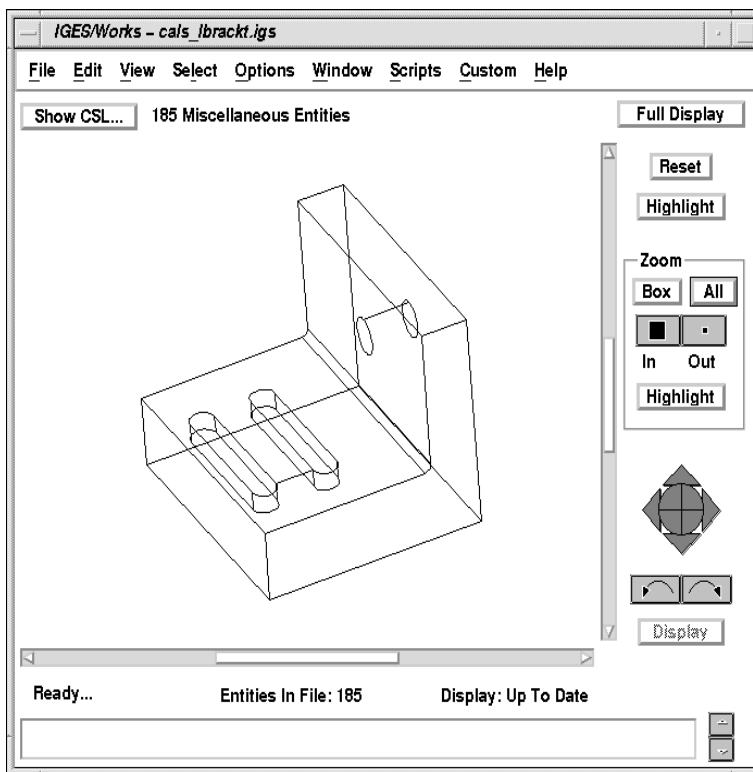
*A detailed explanation of all the configuration settings can be found in the file "config.tpl", located in the "install" subdirectory of IGES/Works.*

## Manipulating the File View

### About the Display Manipulation Controls

IGES/Works provides a number of file display manipulation buttons in the File window. They are arranged in the following categories:

- zoom control
- rotation control
- scroll bars
- general display control.



**The zoom control buttons allow you to** zoom in and out of the file display, and to specify the zoom factor. Click Zoom and drag a box around the area into which you want to zoom.

**The rotation control buttons allow you to** rotate your view around the display X, Y, and Z axes. The rotation amount varies with the distance of the click from the center of the control button.

**The scroll bars allow you to** scroll your view horizontally and vertically.

**The general display control buttons allow you to** make the display fill the whole window, reset the display zoom and rotation settings to the file's original parameters. For example, if you modify the display (by scrolling or zooming), you can reset it back to where it was before you started.

### **Note:**

*The Display button on the File window is enabled once the display no longer matches the stored contents of the file. Actions such as entity creation, deletion or modification can cause the mismatch.*

*When you click the Display button, the file displays using the stored data and the current display options (to change the display options, use the File Menu Display item).*

## **Printing a file**

IGES/Works allows you to print the contents of the display window. The printing can be output either to a file or directly to a printer. The way this functionality is platform dependent.

### **How do you print a file?**

You can print a file by

- choosing print from the File Window, File menu
- running a script

**When you choose print from the File menu in the File window...**IGES/Works prompts you to specify printing options such as orientation, size etc.

**Running a script ...** you can create a script that configures things the way you want and then uses the 'Plot Display' command to print.

## **Validating a file**

### **What is validation?**

IGES/Works' validation{XE "Validation"} tool automatically checks and corrects the validity of entities in an IGES file against explicit and implied IGES rules.

**You can set validation to:**

- varying degrees of thoroughness for entity checks
- perform entity fixes

**You can get these reports:**

- status reports{XE "Reports:validation status"}
- summary reports{XE "Reports:validation summary"}

The validation messages can also be sent to a log file{XE "Log file:of validation messages"}. These messages describe the problem encountered with a given entity. If the automatic fixups are enabled, the validation messages also describe the fix performed on the invalid entity. In addition to the validations is an available summary statement that lists all the IGES types and forms in the file, how many entities of each kind there were, how many were in error, and how many were fixed in the validation process.

**Validation**{XE "Validation:levels of entity problems"} works on two levels of entity problems:

1. **Warning** – entity problems that violate IGES rules, which may result in corrupted or missing entities after translation, but which probably will not translate.
2. **Error** – entity problems that violate IGES rules, which may be extensive enough to prevent a translator from producing an output file.

**Examples of Validation**{XE "Validation:examples of checks"} checks:

- **Warnings**
  - the start and ending radii of a circular arc (100:0) being equal
  - pointers point to compatible and valid entities.
- **Errors**
  - discontinuities in curved lines, surfaces, and trimmed surfaces
  - the start or end points not on the conic.

**Examples of**{XE "Validation:examples of fixups"} **fixups**:

- changing a value that is outside the valid range to a default value (e.g., changing a color {XE "Color:changing"} number to 1 (black) that was outside the valid range of 0-8).
- changing a pointer to null that points to an invalid entity.
- making unequal radii equal for a circular arc.

## Why validate a file?

In some cases you might want to make your own diagnosis and design your own cure for a sick file. In others, you might want to use validation to automatically diagnose and fix the file. Validation {XE "Validation:bugs and IGES specification implementation"} can help when the following diseases appear to afflict the file:

- bugs – the file contains a case (valid or not) a translator did not test or judged unworthy of accounting for
- the IGES specification was implemented incorrectly

See “Ways to use IGES/Works” in Chapter 2, “Introduction to IGES/Works” for more information on choosing the right tool.

## How do you Validate a File?

You can validate{XE "Validation"} a file by

- choosing Validate from the File window's File menu
- choosing Validate from the File menu Open options in the Main window
- running a script

**When you choose Validate from the File window's File menu ...** IGES/Works allows you to set switches that determine

- the extent of the validation – Perform Slow Checks, Make Fixes, Delete{XE "Validation:deleting unfixable entities:setting switches"} Unfixable Entities{XE "Deleting entities:unfixable entities in validation"}
- information about the validation – Show Messages, Show Summary, Create Log File.

**When you choose Validate from the File menu Open options in the Main window ...** you can set Validation {XE "Validation:effect range"} to take effect on

- the next file you open
- all files you open during the current session
- the configuration file

**Note:**

*If you choose Configuration File {XE "Configuration file:setting automatic validation option "} as the range over which you want automatic validation to take effect, the next time you start IGES/Works with the same configuration file, it will always automatically validate files when you open them.*

**Running a script ...** you can write a variety of scripts that tailor validation {XE "Validation:tailoring by using scripts"} to suit your needs in different situations. For example, you could use existing scripts, or write your own scripts that automatically

- create log files {XE "Log file:in scripts to store validation messages"} to store all the validation messages
- set the validation switches
- highlight problem entities (see Select\_Invalid.scr in Appendix B, "Scripts.")
- highlight entities changed by the fix-ups (see Select\_Fixed.scr in Appendix B, "Scripts.")
- combine validation {XE "Validation:using in a script"} with other functions.

See Chapter 12, "Writing scripts" for detailed information on writing the scripts to perform these tasks.

## What do the Validation and Fix-up Messages Look Like?

The following sample shows the kind of validation{XE "Validation:messages"} and fix-up messages you might see either in your log file{XE "Log file:validation messages"} (if you specified one), or in the Main window.

```
***** Key to Validation Level Descriptions *****
```

```
Severity 4 - Explicit Error
```

```
Severity 3 - Implicit Error
```

```
Severity 2 - Explicit Warning
```

```
Severity 1 - Implicit Warning
```

```
***** Entity Validation *****
```

```
===== Entities Which Were Corrected =====
```

```
*** Severity 4 (IEVM_BAD_START_POINT_104) ***
```

```
The start point for this Conic Arc entity (104) is not on the conic.
```

```
Start point value found was %.7e, %.7e.
```

```
    Type:Form      DE #
```

```
    -----      ----
```

```
    104:1          2759
```

```
Action taken: The start point has been moved %.7e units, from %.7e, %.7e
```

```
to %.7e, %.7e.
```

```
This occurred on 1 entities
```

```
*** Severity 4 (IEVM_BAD_END_POINT_104) ***
```

```
The end point for this Conic Arc entity (104) is not on the conic. Start
```

```
point value found was %.7e, %.7e.
```

```
    Type:Form      DE #
```

```
    -----      ----
```

```
    104:1          2015
```

Action taken: The end point has been moved  $%.7e$  units, from  $%.7e$ ,  $%.7e$  to  $%.7e$ ,  $%.7e$ .

This occurred on 1 entities\*\*\*

\*\*\* Severity 4 (IEVM\_BAD\_CONTINUITY\_112) \*\*\*

This Parametric Spline Curve entity's (112) Degree of Continuity (Index 2 of the Parameter Data) is incorrectly specified as  $%d$ . Degree of Continuity is calculated to be  $%d$ .

Type:Form	DE #
-----	----
112:0	945, 1381, 1437, 1441, 1491, ...

Action taken: The Degree of Continuity has been set to  $%d$ .

This occurred on 15 entities

==== Entities Which Were Not Corrected =====

\*\*\* Severity 2 (IEVM\_BAD\_JUSTIFICATION\_212) \*\*\*

This General Note entity's (212:6, 7, or 8) substrings are not correctly justified relative to the entity's form number.

Type:Form	DE #
-----	----
212:6	555, 1099, 1497, 1497, 1497
212:8	557, 573, 825, 1407, 2887

This occurred on 10 entities

\*\*\*\*\*

**Note:**

*You would only get these messages if you set the quiet/verbose switch to verbose (which is the default).*

The following is a sample summary report{XE "Reports:validation summary sample"}.  
 Entity Validation Summary:

Type	Form	Entity Count	Number Valid	Number of		Number of	
				Corrected Warnings	Errors	Uncorrected Warnings	Errors
Global Section		1	1	0	0	0	0
0	0	10	10	0	0	0	0
100	0	242	242	0	0	0	0
104	1	15	13	0	2	0	0
106	11	115	115	0	0	0	0
106	21	6	6	0	0	0	0
106	31	38	38	0	0	0	0
106	40	44	44	0	0	0	0
110	0	631	631	0	0	0	0
112	0	16	1	0	15	0	0
124	0	15	15	0	0	0	0
202	0	8	8	0	0	0	0
206	0	8	8	0	0	0	0
210	0	8	8	0	0	0	0
212	0	152	152	0	0	0	0
212	6	67	64	0	0	5	0
212	7	1	1	0	0	0	0
212	8	10	5	0	0	5	0
214	2	88	88	0	0	0	0
214	4	2	2	0	0	0	0
216	0	14	14	0	0	0	0
222	0	2	2	0	0	0	0
228	1	14	14	0	0	0	0
228	3	38	38	0	0	0	0
402	7	2	2	0	0	0	0
406	7864	13	13	0	0	0	0

-----  
Totals: 1560 1535 0 17 10 0

## How do you Interpret the Validation Messages?

The example below was extracted from the sample messages{XE "Validation:interpreting the messages"} shown on the previous page.

```
*** Severity 4 (IEVM_BAD_START_POINT_104) ***

The start point for this Conic Arc entity (104) is not on the conic. Start
point value found was %.7e, %.7e.

Type:Form      DE #
-----      ----
104:1          2759

Action taken: The start point has been moved %.7e units, from %.7e, %.7e to
%.7e, %.7e.

This occurred on 1 entities
```

This message has the following four parts:

- Header
- Message
- Prefix
- Fix (when applicable)

**The header ...** \*\*\* Severity 4 (IEVM\_BAD\_START\_POINT\_104) \*\*\* identifies the condition as a Warning or Error, and contains the error code in parentheses. The error code is the unique identifier for this particular validation check (there are over 400 total). Reference the error code in the IGES/Works external file "validation.hlp", located in the help subdirectory, for a detailed explanation.

**The message ...** The start point for this Conic Arc entity (104) is not on the conic. Start point value found was %.7e, %.7e. gives details of the check.

```
The prefix ... Type:Form      DE #
                -----      ----
                104:1          2015
```

identifies the Directory Entry (DE) number, and the type and form of the IGES entity.

**The fix ...** Action taken: The start point has been moved %.7e units, from %.7e, %.7e to %.7e, %.7e. This occurred on 1 entities displays only when the Validation has found an error and has performed a fix on the entity.

### Note:

*Validation modifies the file in memory if you request fixes. If you want to keep the original file as it was, save your validation work under another file name.*

## ***Projecting a File***

### **What is Projection?**

Projection is the process of “flattening” a 3D file into its 2D equivalent by *projecting* it through a specified view or drawing. It is one way to simplify an IGES file.

### **Why Project a File?**

IGES files often contain structures that are too complex for the target system. Projection is one of the IGES/Works tools you can use to simplify the file{XE "Projection:tool for simplifying a file"}. See “Ways to use IGES/Works” in Chapter 2, “Introduction to IGES/Works” for more information on choosing the right tool.

### **How do you Project a File?**

You can project a file by

- choosing Project from the File window, File menu
- running a script

***Choosing Project from the File menu in the File window ...*** IGES/Works prompts you to specify the view (and other projection parameters) through which you want to project the file.

***Running a script ...*** you can run an existing script named Project\_IGES.scr, or create your own scripts that specify projection{XE "Projection:using in scripts"} views and angles, and combine this process with other functions.

### ***Working with the Start Section of a File***

The Start section contains general information about the file's contents and how it was created. This data may be useful when you are analyzing the file.

You can perform the following operations on the Start section:

- list its contents
- copy it between files
- extract one or more lines
- modify its contents

## Listing the Start Section

You can list the Start section of a file by choosing List Start from the File window File menu

## Copying the Start Section

Copying a Start section is useful when you are creating new IGES files, or overwriting the existing Start section of an IGES file.

You can copy the start section of a file by choosing Copy, then Start Section from the File window Edit menu.

## Extracting Data from the Start Section

You can extract one or more consecutive lines from the Start section into an IGES/Works variable and format it into a report{XE "Reports:extracting lines from Start Section"}. You might want to do this in a script.

You can use the *get start\_field* command to retrieve information on the Start section in the current file. See the *get start\_field* command in the *IGES/Works Command Reference Manual* for details.

## Modifying Data in the Start Section

You can modify the Start section by using the *modify start\_section* command. See the *modify start\_section* command in the *IGES/Works Command Reference Manual* for details.

## Working with the Global Section of a File

The Global section contains general information about the file's contents.

You can perform the following operations on the Global section:

- list its contents
- copy it between files
- extract one or more field values
- modify its contents

## Listing the Global Section

You can list the Global section of a file by choosing List global from the File window File menu.

## Copying the Global Section

Copying a Global section is useful when you are creating new IGES files, or overwriting the existing Global section of an IGES file.

You can copy the Global section of a file by choosing Copy, then Global Section from the File window Edit menu.

## Extracting Data from the Global Section

You may extract one or more field values from the Global section into an IGES/Works variable. See the *get global\_section* command in the *IGES/Works Reference Manual* for details.

## **Modifying the Global Section**

You can modify any of the fields in the Global section by choosing Modify, then Global Section from the File window Edit menu.

## Chapter 5. Working with Entities

*In this chapter you learn about*

- selecting entities
- getting more information about entities
- copying entities
- editing entities

### **Selecting Entities**

Some IGES/Works commands and functions work on the file as a whole. Others just work on specific entities you select. Before you can work with, or get more information about an entity you must first select it – and by doing so, place it in the Current Selection List.

### **What is the Current Selection List?**

IGES/Works allows you to make a list of the entities you select{XE "Selecting entities:making a list of them"}. The Current Selection List (CSL) is the selection list you are currently working with. You can save this list and use it later, and meanwhile put a whole new selection of entities in the CSL. A saved selection list that you restore becomes the Current Selection List. See “Manipulating selection lists from the command line” below, for more information about this.

### **Ways to Select Entities**

IGES/Works offers a variety of ways you can select entities{XE "Selecting entities:variety of methods"}:

- by clicking on entities in the display
- by using the File window Select pulldown menu
- by using the Selection List window
- from the command line

**Clicking on entities in the display ...** allows you to select entities in the following ways:

- single clicking on an entity selects only that individual entity
- CTRL clicking **adds** an individual entity to the selection list
- SHIFT clicking makes a hierarchical selection (starting at the bottom). For example: A single click selects an arc. Whereas, a shift click on the same entity selects its parent, a composite curve.

#### **Note:**

*If the selected entity has no parents, you will hear a beep. The Current Selection List will be left intact.*

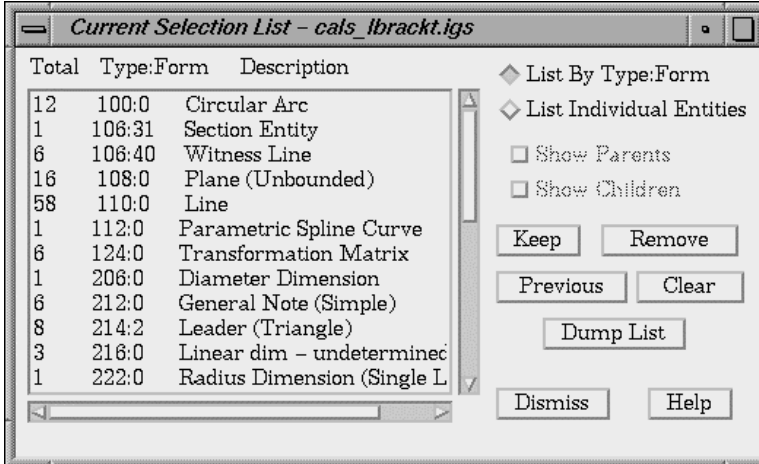
- pulling a box selects all entities in the box

**Using the File window Select pulldown menu ...** you can select

- entities by DE number
- all the entities in the file

- all the independent entities in the file
- all the child entities of the current selection
- all the parent entities of the current selection
- by DE field values.

For example: Select all green entities on level 12.



**Using the Selection List window ...** you can manipulate the Selection List by

- using the Keep button to keep entities you highlight in the list (removing the others)
- using the Remove button to remove entities you highlight in the list (keeping the others)
- showing parents and children of entities in the CSL
- using the Previous button to restore the previous CSL.

**From the command line ...** you can use the *Find Entity* and *Change Selection* commands to select the entities{XE "Selecting entities:using the *find entity* and *change selection* commands"} you want to work with. You may also want to become familiar with this technique if you need to write scripts. See Chapter 8, "Understanding the Data Definition File," and Chapter 9, "Navigating the IGES entity hierarchy" for more information about using these commands. You can also use the *Select Children* command. See *The Command Reference Manual* for more information about using the *Select Children* command.

Exactly how you go about selecting the entities{XE "Selecting entities:variety of methods:using in combination"} you want to work with depends on how you prefer to work and what your goal is. You can use the various methods described above in combination to get the selection of entities you need.

### Tip

*You could use the Selection pulldown menu to place a group of entities in the Current Selection List, then use the CSL window buttons to remove the individual entities that you do not need from the list.*

## Manipulating selection lists from the command line

There are two related IGES/Works commands; *save selection* and *restore selection*. The *save selection* command will write the contents of the Current Selection List to a saved selection list. For example, suppose that the Current Selection List contained all of the lines in the model and you wanted to save this list for future reference. The following command performs this operation.

```
save selection name all_lines
```

After some intermediate operations, you now want to perform additional analysis on the line entities in the model. You can retrieve the saved selection list of line entities and place them in the Current Selection List with the following command.

```
restore selection name all_lines
```

There are four selection list manipulation operations accessible through various IGES/Works commands; *add*, *diff*, *intersect*, and *merge*. These operations are useful in redefining or creating a specific set of entities in a selection list.

The *add selection* command adds the contents of the Current Selection List to a saved selection list. Through the command's switches, you may maintain hierarchical lists or ignore this list attribute.

For example, the following command adds the Current Selection List to the saved selection list named 'border' ONLY if the two lists share the same hierarchy.

```
add selection name border -hierarchy
```

The *diff selection* command searches two saved selection lists for entities that are NOT shared and places the differing entities in the Current Selection List. Through the command's switches, you may maintain hierarchical lists or ignore this list attribute. For example, the following command places in the Current Selection List the entities from the saved selection lists named 'list\_a' and 'list\_b' that are not in common between the two lists, regardless of the lists' hierarchy attribute.

```
diff selection name1 list_a name2 list_b
```

The *intersect selection* command will determine the intersection of two saved selection lists and place the intersecting entities in the Current Selection List. The hierarchy status of the Current Selection List will be the same as the status of the saved selection list specified first in the command. An example is given below.

```
intersect selection name1 ab_ent name2 bc_ent
```

The *merge selection* command will combine two saved selection lists into the Current Selection List. Through the command's switches, you may maintain hierarchical lists or ignore this list attribute. In the following command the contents of the two saved selection lists are placed in the current selection list only if the two lists share the same hierarchy.

```
merge selection name1 lines name2 circles -hierarchy
```

## Getting more Information About Entities

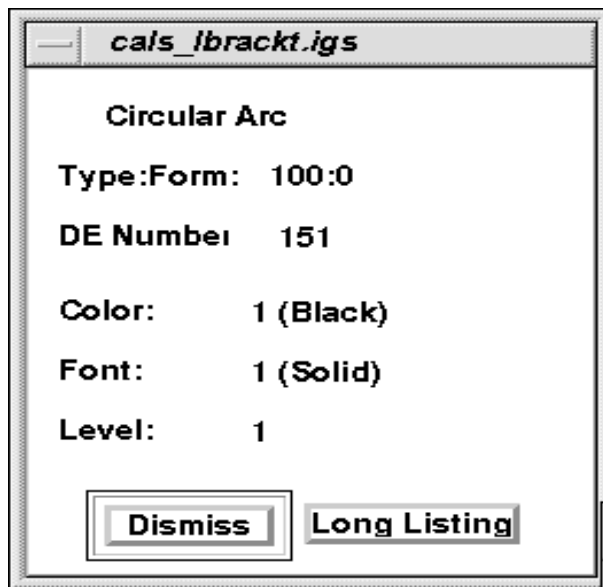
You can get more information about an entity by listing its fields. You can do this by

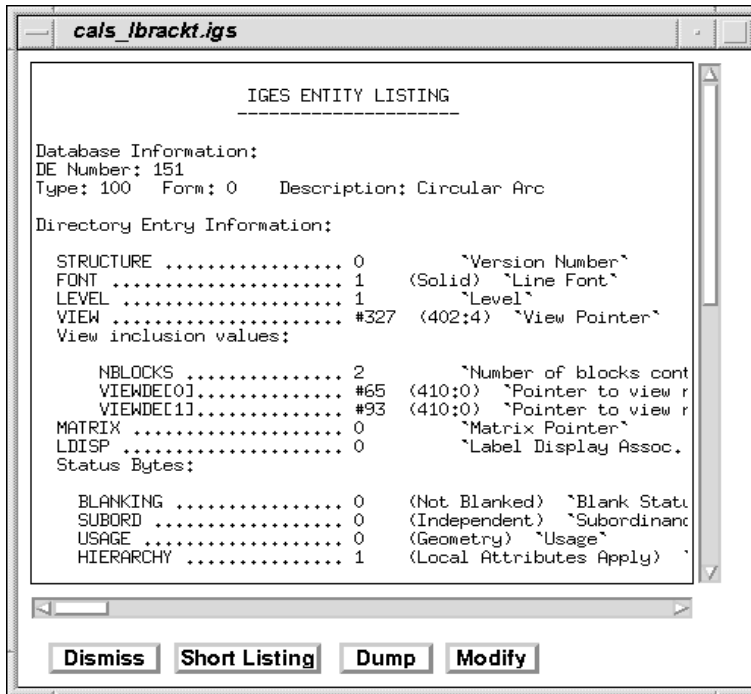
- double clicking on the entity in the display to get an Entity Listing dialog{XE "Dialog:Entity Listing:getting more information about an entity"}{XE "Entity Listing:getting more information about an entity"}
- double clicking on an individual entity in the CSL to get an Entity Listing dialog

IGES/Works offers a short and a long entity listing. The default is the short listing. During a session, you can switch to the long (more detailed) listing. You can change this default by using the Options menu or in the configuration file{XE "Configuration file:changing Entity Listing default"}.

### Tip

*You can also highlight the entity in the Selection List and choose Dump List (from the long Listing dialog) to send the entity listing out to the text display in the main window. This way, you can save it to a Log File{XE "Log file:saving an entity listing"}, or copy it into a paste buffer{XE "Main window:using the text display area"} and use it in building a document or report{XE "Reports:using an entity listing"}.*





Once you have selected the entities you want to work with, you can

- edit them
- convert them
- run a script on them

### **Copying Entities**

An option is provided for copying entities to the same file. A modified dialog box is provided.

When copying is being done to the same file, it shows two options:

1. Create New Associated Entities
2. Keep Existing DE Entity Pointers

### **Editing Entities**

The editing tools{XE "Editing entities:Edit menu options"} are located in the Edit pull-down menu on the File window. These tools (except for Create{XE "Creating entities:editing tools"}) work on the entities you place in the Current Selection List or with the whole file. They allow you to

- copy the entities to another file
- delete the entities
- modify the entity
- create new entities
- convert the entities

**When you choose copy ...** IGES/Works prompts you with a file dialog for the name of the file to which you want to copy the entities{XE "Copying entities:file open dialog"}{XE "Copying entities:file open dialog"} you selected. This can be a new (empty) file or an existing file.

**When you choose delete ...**{XE "Deleting entities: no undo command"}{XE "Deleting entities: no undo command"} the entities you selected are deleted from the file in memory. If you save this file

under its original name the entities you deleted will be lost. If you need to keep the original file as is, you should save your work under a different filename.

**Note:**

*IGES/Works does not store entities when you delete them, and there is no “Undo” command. So, you cannot delete entities and then bring them back by pasting them somewhere else or by undoing the delete.*

**When you choose modify ...** IGES/Works provides you with a dialog{XE "Dialog:modifying an entity"}{XE "Modifying entities"} through which you can change the Entity Description (DE Section), and the Parameter Data Section for the entity you selected.

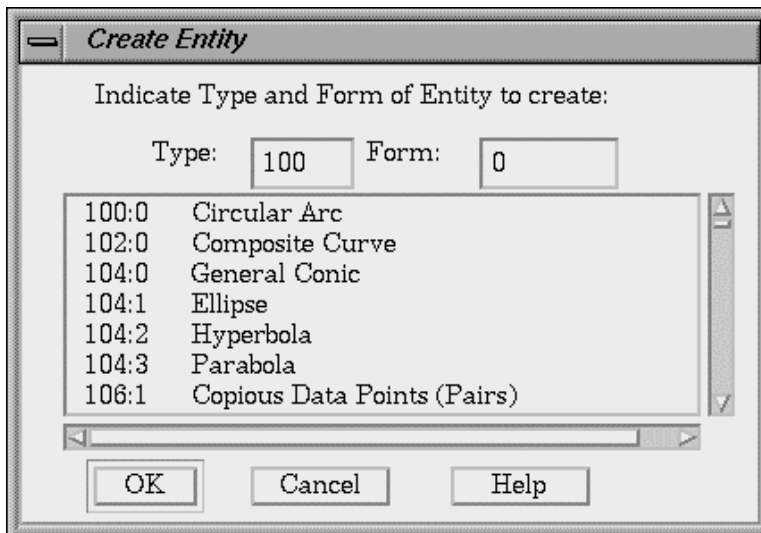
The dialog box is titled "Modify DE Values". It has a section for "Entity Description" with fields for "Type:", "Form:", and "DE:". Below this are several parameters, each with a checkbox and a value field:

- Structure:** 0 (Value/Pointer icons)
- Line Font:** 1 (Solid) (Value/Pointer icons)
- Level:** 0 (Value/Pointer icons)
- View:** 0
- Matrix:** 0
- Label Display:** 0
- Blanking:** 0 (Not Blanked) (Value/Pointer icons)
- Subordination:** 0 (Independent) (Value/Pointer icons)
- Entity Use:** 0 (Geometry) (Value/Pointer icons)
- Hierarchy:** Attributes Apply (Value/Pointer icons)
- Weight:** 0
- Color:** Color Assigned (Value/Pointer icons)
- Label:** (empty field)
- Subscript:** 0

At the bottom of the dialog, there is a "Modify PD Fields" button and four standard buttons: "OK", "Reset", "Cancel", and "Help".

Since the DE section is common to all entities you can select and modify multiple entities{XE "Modifying entities:multiple selection"} by DE section. For example, if you want to make all entities green, this is the way to do it. The PD section button is disabled when you select multiple entities.

**When you choose create ...** a Create Entity dialog{XE "Creating entities:Create Entity dialog"} appears with a list box displaying the entities you can create. Pick one and IGES/Works will create that entity with default values that you can later modify.



**When you choose convert ...** IGES/Works converts the entities you select into IGES equivalents. See Chapter 6, "Working with conversions" for more information.

## Chapter 6. Working with Conversions

*In this chapter you learn about*

- converting entities
- trimmed surface fix-ups
- reversing surface normals

### **Converting entities**

IGES/Works provides conversion functions to change the entities you select into equivalent IGES entities. You can also perform fix-ups on curves for Trimmed Surfaces and reverse surface normals.

### **Why convert entities?**

Conversion allows you to make the entities you select more compatible with the target system. Using the conversion modifiers and switches, you can perform a wide variety of entity conversion and preparation tasks. See “Ways to use IGES/Works” in Chapter 1, “Introduction to IGES/Works” for more information on choosing the right tool.

### **How do you convert entities?**

Select the entities you want to convert, then do one of the following:

- choose one of the items from the Conversion submenu (off the Edit menu in the File Window)
- run a script

### **How do you fix Trimmed Surface entities?**

Select the entities you want to convert, then do one of the following:

- choose Regenerate Trim Curves from the Fix sub-menu off the Edit menu in the File Window
- run a script

### **How do you reverse surface normals?**

Select the surfaces you want to reverse, then do one of the following:

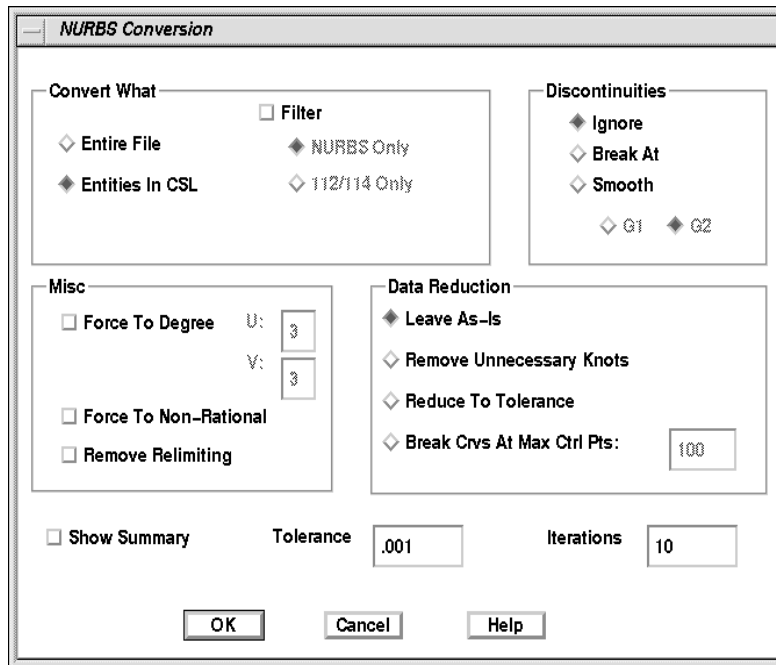
- choose Reverse Surface from the Fix sub-menu off the Edit menu in the File window
- run a script

***When you choose one of the conversion options from the Convert sub-menu off the Edit menu in the File window...*** the various conversion dialogs allow you to specify how you want to convert the entities you selected. You can choose between converting the entities to:

- NURBS curves and surfaces (Types 126 and 128)
- parametric spline curves and surfaces (Types 112 and 114)
- linear segments (Type 106 Form 12)
- Trimmed Surfaces (Type 144). This can only be done to Bounded Surfaces (Type 143). If the 143s have child entities that are anything but NURBS-based, they are

converted to NURBS curves and surfaces. This dialog does not, however, convert independent 141s to 142s.

- (NURBS to their analytical form (if possible) i.e. Arcs, Lines, Ruled Surfaces, etc.).



On the NURBS Conversion dialog you can also set various options.

- Force to Degree – To specify the maximum degree in the equations that define the NURBS curves and surfaces.
- Force To Non-Rational – To force all processed curves and surfaces to be non-rational.
- Remove Relimiting – To remove any relimiting from the processed curves and surfaces.
- Discontinuities – Discontinuities can either be ignored or smoothed over at points of G1 or G2 discontinuities. Curves and surfaces can also be broken up or divided at the points of discontinuity.
- Data Reduction – Data in NURBS curves and surfaces can be reduced by removing unnecessary knots, or by approximating to a tolerance or by dividing curves that exceed a maximum number of control points.
- Iterations – The maximum number of times the conversion algorithms are performed to achieve the most accurate results. The default is 10 iterations.
- Tolerance{XE "Tolerances:in the Convert dialog"} – To specify how close the resultant entity converges with the original. The default is .001.
- Show Summary – This option will output a short summary of the entities processed.

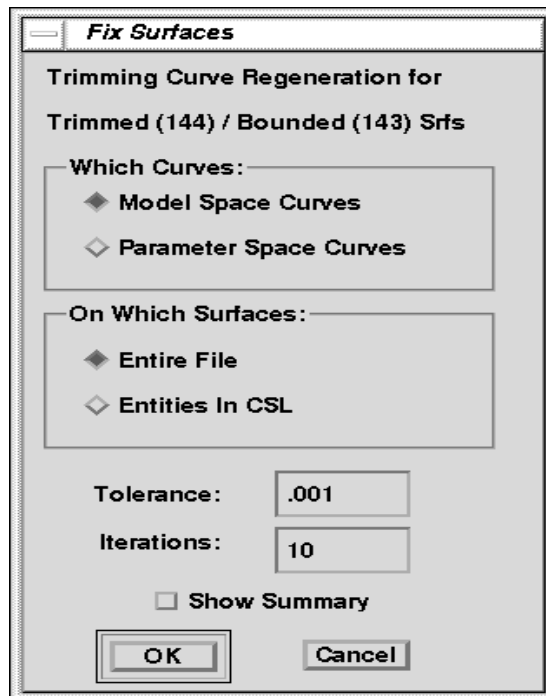
Other conversion dialogs have some subset of these options. NURBS conversion is the most fully featured conversion.

**Note:**

*When converting to Trimmed Surfaces, the Degree values are operative only if the Bounded Surface entity has an offset (130) that points to a parametric or NURBS curve. In this case, a new NURBS curve will be approximated. This approximation mechanism enables you to specify the maximum degree for the resulting entity, and the number of iterations the algorithm performs to achieve an accurate approximation. For all other children of Trimmed or Bounded Surfaces, the Degree values do not apply.*

**When you choose Regenerate Trim Curves form the Fix sub-menu off the Edit menu in the File window...** the dialog{XE "Dialog:Fix"} prompts you to specify one of the following:

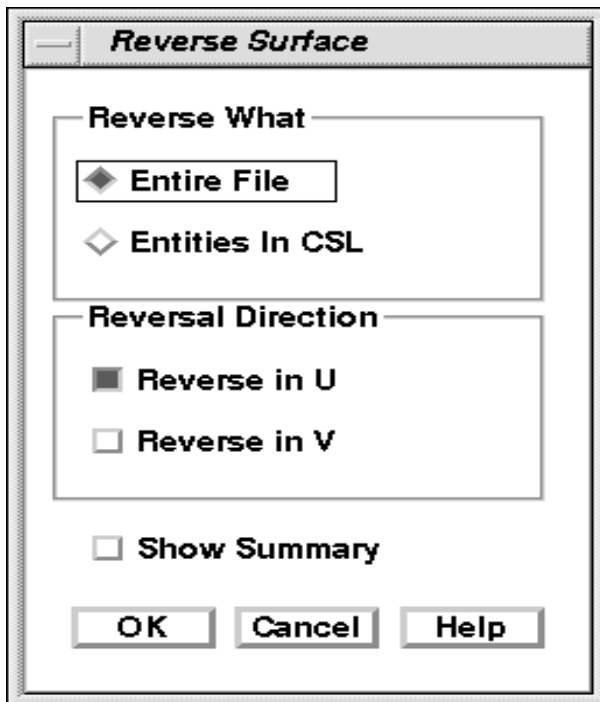
- Model Space Curves – Generate the model space form of the trimming curves for Trimmed Surface entities (144 only). The underlying geometry must be NURBS-based (types 126 and 128) or this command is not applicable.
- Parameter Space Curve – Generate the parameter space form of the trimming curves for Bounded Surface and Trimmed Surface entities (143 and 144, respectively); converts supporting curves and surfaces to NURBS.



You can also use this dialog {XE "Dialog:Convert:setting modifiers and switches"} to set various conversion modifiers and switches.

- Iterations – The maximum number of times the conversion algorithms are performed to achieve the most accurate results. The default is 10 iterations.
- Tolerance{XE "Tolerances:in the Fix dialog"} – To specify how close the resultant entity converges with the original. The default is .001.
- Show Summary – This option will output a short summary of the entities processed.

**When you choose Reverse Surface from the Fix sub-menu off the Edit menu in the File Window...** the dialog allows you to reverse the U parametric direction, the V parametric direction or both. Choosing only one effectively reverses the surface normal.



**Running a script ...** You can automate entity conversions by making them part of a script. For example, you could write a script that selects certain entities{XE "Selecting entities:by using scripts"}, converts them with certain modifiers and switches, and then outputs a log file {XE "Log file:conversion output"}recording the results. See Chapter 7, "Running scripts" and Chapter 12, "Writing scripts," for more information about scripts.

## Chapter 7. Running Scripts

*In this chapter you learn*

- what are scripts?
- why use scripts?
- how to run scripts

### **What are Scripts?**

A script is a text file containing a series of IGES/Works commands. You can use a script in the same way you would a macro in other software packages.

IGES/Works ships with a set of ready made scripts that you can use as is, or modify to suit your needs. See Appendix B for descriptions of all the scripts shipped with IGES/Works. You can also write your own scripts. See Chapter 11, "Writing scripts" for more information about this.

### **Why use Scripts?**

Whereas IGES/Works commands are rather granular in function, scripts allow you to develop entire functions tailored to your specific needs.

Scripts provide a simple way to

- flavor an IGES file
- generate a report{XE "Reports:using scripts to generate"}
- query data
- create IGES entities
- make entity modifications to flavor IGES data for a specific CAD system.

For example, you can create scripts that automatically select certain entities, convert them with certain modifiers and switches and then output a log file recording the results. You can write simple scripts or create a whole system of functions to flavor IGES files to automate the process of adhering to design standards.

## **Running Scripts**

Whether you want to use the scripts shipped with IGES/Works or ones you create, you can run them from

- the Scripts menus
- the Custom menus
- the command line
- inside another script.

### **Note:**

*It is a good idea to put all your scripts in a standard directory – to make them easier to locate.*

## **Running Scripts from the Scripts Menus**

You can run any script from the Scripts menus on either the Main window or File window. This includes Python scripts also (Execute Python Scripts). When you open the Scripts menu, you can choose to either view or execute scripts. Choosing either option takes you to a standard dialog{XE "Dialog:selecting scripts"} box – allowing you to choose from a selection of scripts. View allows you to look at the content of scripts without running them. To run a script, choose Execute and select the script you want. If you are running a script that is taking too long, you can interrupt it by pressing the “Q” key. This will stop processing. If you do this, you may not be able to use some commands or update the display since the model may not be in a valid state. If you experience unpredictable behavior after interrupting the processing, you should close the model.

### **Note:**

*If you have more than one file open, make sure your focus is in the window of the file against which you want to run the script. If you are working in the Main window, the script will run against the current file (the one in which you performed the last operation).*

## Running Scripts from the Custom Menus

You can assign special, or frequently used scripts to the Custom menu on either the Main window or File window. If a script has been assigned to a Custom menu, you can run that script by opening the Custom menu and choosing that item. In some cases, the script name may be listed under a sub-menu from the Custom menu. You can assign any script to the Custom menus on both the File window and Main window by editing the configuration file. See "Editing the configuration file" in Chapter 3, "Working with files."

### Tip

*You can use a script instead of the dialog if you want to configure things differently. Among the scripts shipped with IGES/Works are those that used to run the above mentioned operations. You can use them to create alternatives to the standard dialogs.*

*You could use a script to combine several operations into a single command. For example, selecting and unblinking blanked entities before display.*

## Running Scripts from Inside Another Script

You can run a script from inside another script in the same way you would call a subroutine in many programming languages. You can even pass data between scripts. See "Passing data into the script" in Chapter 11, "Writing scripts."

## Running Scripts from the Command Line

Issue the following command to invoke a script:

```
execute script file name-of-script
```

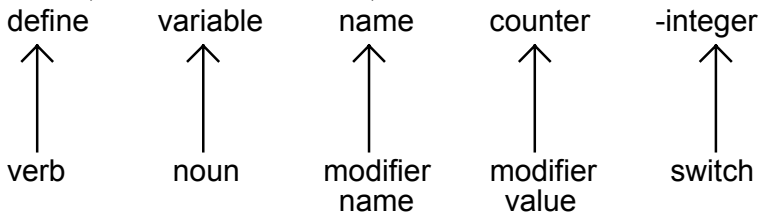
## Chater 8. Entering IGES/Works Commands

In this chapter you learn about

- syntax of IGES/Works commands
- using aliases for IGES/Works commands
- command line features
- working with message output
- working with operating system commands
- using and modifying wild card characters

### Syntax of IGES/Works Commands

Every IGES/Works command {XE Command:syntax:components"} has four components; a verb, a noun, zero or more modifiers, and zero or more switches.



These are explained below. All command components are case insensitive; they may be entered in either upper, lower, or mixed case. Modifiers and switches may appear in any order.

- **verb** – Defines the action that will be performed.
- **noun** – Defines the object upon which the action will be performed.
- **modifier** – Used to add more detail to commands that are broad in scope. Some commands which perform very specific functions have no modifiers. Modifiers conform to a name/value pair. Some modifiers are optional in which case a default value is used. Other modifiers are required as part of the command's syntax.
- **switch** – Used to tailor the operation of a command similarly to modifiers. In contrast to modifiers, switches are a single value and are prefaced by a dash (-{XE "Special character:dash (-):preface to a switch"}).

As an example of the command syntax{XE "Command:using modifiers and switches"}, four different forms of the *list script* command are listed below. Notice how the use of modifiers and switches provide slightly different flavors of the same basic command.

- 1 list script
- 2 list script name Query{XE "Special character:asterisk (\*) :wild card"}\*
- 3 list script -long
- 4 list script name Query\_Fonts -long

### Form 1

The basic command which will list the names of all scripts in the pre-defined script directory.

## Form 2

Depicts the use of a modifier. Note the name/value pairing in which the modifier name is 'name' and the value is 'Query\*'. It will list the names of all scripts in the pre-defined script directory that begin with 'Query'.

## Form 3

Depicts the use of a switch. Note the dash (-) before the switch name 'long'. There should be no blanks between the dash and the switch name. This command will list the headers of all scripts in the pre-defined script directory.

## Form 4

Is an example of the simultaneous use of modifiers and switches. This command lists the header associated with the script 'Query\_Fonts' in the pre-defined script directory.

There are various ways that a modifier value may be expressed which are described below.

A value may be an explicit integer (such as 3, -87, 45).

A value may be an explicit real (such as 92.3, -0.04, 0.45E04).

A value may be an explicit character string. If the string contains embedded blanks, it must be enclosed within parenthesis. Some examples are listed below.

```
echo data args Hi!
```

```
echo data args (Hello, today is Friday)
```

If a character value contains a dollar sign '\$', an asterisk '\*', or a question mark '?', it must be enclosed in quotes because these characters have special meaning. Alternatively, these characters may be preceded by a back slash '\' rather than enclosing the entire string in quotes.

```
echo data args ("What is your name?")
```

```
echo data args (What is your name\?)
```

A value may be a variable. Variables used as the value of a command modifier must be preceded with a dollar sign '\$'. See Chapter 10, "Working with IGES/Works variables."

```
echo data args $date
```

If an IGES/Works variable is an array, each subscript is enclosed within square brackets '[, ]'. See Chapter 10, "Working with IGES/Works variables."

```
echo data args $pnts[3][4]
```

An array subscript may be an explicit integer or an IGES/Works integer variable.

```
echo data args (Today is $day[4])
```

```
echo data args (Today is $day[$ndx])
```

If a value is composed of multiple tokens, the tokens are separated by commas.

```
build script args ($a1, $a2)
```

## Using Aliases for IGES/Works Commands

Aliases {XE "Aliases:assigning names or synonyms "}are a means by which you assign a name or synonym to a particular function. The function can be a command or a portion of a command. Aliases are used to shorten the amount of keystrokes required to execute a function or to equate a mnemonic that is meaningful to you.

As an example, you may frequently execute scripts within IGES/Works. The command for this is

```
execute script name name-of-script.
```

You could define an alias{XE "Aliases:defining "} of `es` for "execute script name" so that instead of typing the entire command, you could enter

```
es name-of-script
```

Similarly, if you wanted to exit IGES/Works, you would enter the command

```
quit system -force
```

However, you could create an alias{XE "Aliases:creating "} named "bye" which would perform the same function. Now you could enter `bye` to exit IGES/Works.

The IGES/Works *define alias* command creates aliases. The alias definitions for the two examples in the previous paragraphs are listed below. Please refer to the *IGES/Works Command Reference Manual* for additional information on the *define alias* {XE "Aliases:defining:define alias command "}command.

```
define alias name es value (execute script name)
```

```
define alias name bye value (quit system -force)
```

If you define aliases that you want available each time you use IGES/Works, place the alias definitions in the start-up script file or the configuration file{XE "Aliases:defining:in start-up script file "}. This way, you are guaranteed of having them defined in every IGES/Works session.

## Command Line Features

There are a few useful features when entering commands on the command line.

- GUI arrows, found at the end of the command line, cycle previously issued commands
- "{XE "Special character:exclamation marks (!!):when entering commands on the command line"}" with or without additions, re-issues the previous command (similar to the UNIX command line)
- **Example:** If the previous command was:  

```
list script
```

entering this:  

```
!! -long
```

would issue the following command:  

```
list script -long.
```
- "{XE "Special character:carets (^):when entering commands on the command line"}" alters the previous command via a "search-for" string and a "replace" string (similar to the UNIX command line).

- **Example:** If the previous command was:  
 echo data args (Hi Heidi)  
 entering this:  
 ^Hi^Hello  
 would issue the following command:  
 echo data args (Hello Heidi)

## ***Working with Message Output***

IGES/Works messages are issued to inform you of some condition which exists or unexpected event which occurred. There are many different reasons why a message may be issued but the main purpose is to inform you of what is going on within the software.

Messages are divided into four classifications: status, informational, warning, and error. By default, output from all message classes are directed to the terminal. However, you may re configure message output in a variety of ways. Separate control is provided for each class. The message output configuration is accessed through the *redirect messages* command. Refer to the *IGES/Works Command Reference Manual* for a complete description of this command. The modifier 'file' defines the file name which will receive all output from the particular message classes specified in the command.

There are three message output device choices: the terminal, a file, or both the terminal and a file. The switches: -terminal, -file, and -both, respectively, control the output destination. There are five message class switches: -status, -info, -warning, -error, and -all. If any of the message classes are directed to a disk file, then the switches '-command' and '-no\_command' may be specified. These switches control whether or not the command line contents are written to a file.

In the *redirect message* command, you may mix and match the output device and the message class. And if the file modifier is specified, you can also specify the command line output.

Below are a few examples of message redirection. It is by no means an exhaustive list. It is a guide for you to create your own message output configuration.

The following command outputs all status and informational messages to a file named 'nice\_to\_know.msg'.

```
redirect message -status -info -file file nice_to_know.msg
```

The following command outputs all warning and error messages and all command line entries to a file named 'must\_read.msg'.

```
redirect message -warning -error -command -file \  
file must_read.msg
```

The following command outputs the error messages to the terminal and to a file named 'all\_error.msg'.

```
redirect message -error -both file all_error.msg
```

## Working with Operating System Commands

You may execute an operating system command directly from IGES/Works. You may want to list the names of IGES files in one of your directories or invoke a process of some kind. If the operating system command results in a text display, it appears in the IGES/Works Main window. The *invoke process* command issues operating system commands. Examples from the UNIX environment are listed below.

```
invoke process args ("ls")

invoke process args ("cat /users/test/short_file.igs")

invoke process args ("rm bad_data.igs")
```

### Note:

*If the command is greater than one token, the command must be enclosed within double quotes ("").* {XE "Special character:double quotes (\\" \"):when entering commands on the command line"}. The IGES/Works Command Reference Manual describes this command in further detail.

## Using and Modifying Wild Card Characters

Some IGES/Works commands allow a wild card character as part of a modifier's value. There are six wild cards {XE "Wild cards"} in IGES/Works which are listed and described below.

- `wc_multi_place` A place holder for 0, 1, or more alphanumeric characters. Default = `*` {XE "Special character:asterisk (\*)as a wild card"}.
- `wc_single_place` A place holder for 1 alphanumeric character. Default = `?` {XE "Special character:question mark (?)as a wild card"}.
- `wc_multi_alpha` A place holder for 0, 1, or more alphabetic characters. Default = `a` {XE "Special character:A (the letter)as a wild card"}.
- `wc_single_alpha` A place holder for 1 alphabetic character. Default = `A` {XE "Special character:a (the letter)as a wild card"}.
- `wc_multi_numeric` A place holder for 0, 1, or more numeric characters. Default = `n` {XE "Special character:N (the letter)as a wild card"}.
- `wc_single_numeric` A place holder for 1 numeric character. Default = `N` {XE "Special character:n (the letter)as a wild card"}.

Several examples of wild cards {XE "Wild cards:examples"} are given below.

The following command lists all scripts that begin with the word 'List'.

```
list script name List*
```

The following command list all variables that have a name containing exactly three characters (any combination of alphabetic and numeric characters).

```
list variable name ???
```

The following command tests the variable named 'part\_name' to see if it conforms to the syntax {XE "Command:syntax:conforming to"} of three alphabetic characters followed by four numeric characters.

```
if condition ("aaannnn",eg,$part_name)

    echo data args (Part name is valid.)

end_if
```

The following command tests the variable named 'part\_num' for conformance to the syntax of 1 or more numeric characters followed by 1 or more alphabetic characters.

```
if condition ("nNaA",eq,$part_num)

    echo data args (Part number is valid.)

end_if
```

You may modify the default setting of the six wild card characters through the *modify config* command. Refer to the *IGES/Works Command Reference Manual* for additional information.

## Chapter 9. Understanding the Data Definition File

*In this chapter you learn about*

- the Data Definition File representation of the IGES specification and about referencing an entity's
- PD data
- DE data
- property pointers
- derived fields

### **The Data Definition File Representation of the IGES Specification**

With IGES/Works you can create{XE "Creating entities:definitions in the IGES specification"}, read, and write all IGES entities as well as access any specific field in an entity's DE and PD record. To provide this capability the software must know how all entities are defined within the IGES specification.

The Data Definition File (DDF) {XE "DDF"} is a file which contains all the entity definitions and is one of the cornerstones of IGES/Works. It is important to understand the syntax of the entity definitions in order to use several IGES/Works commands such as *change selection*, *find entity*, and *get field*. See Chapter 9, "Navigating the IGES entity hierarchy," for specific details on these commands.

The DDF can be thought of as a computer sensible representation of the IGES specification. To understand how a line entity is defined, a person would read the IGES specification and a computer program would read the DDF file. Every IGES entity is defined by the following four groups of information:

- DE data (common entity display information).
- PD data (unique entity definition data).
- property data (0, 1, or more references to properties or attributes which further define an entity).
- associativity data (0, 1, or more references to associated or associating entities).

Both the DDF and the IGES specification contain this defining data. The DDF contains one extra set of fields – derived fields. These are data values which can be derived from the PD data of the entity.

You can use the *list ddf* command{XE "DDF:list ddf command"} to produce a DDF listing for the Circular Arc entity. For example:

```
list ddf type 100
```

The above command will produce the listing shown on the next page. Notice that the listing contains the data groups listed above. In the 'Entity Fields' portion you find 'igesde' is the entity's DE data, 'props' – the entity's property data, 'assocs' – the entity's associativity data, and the remaining entity fields are the entity's PD data.

In the 'Derived Fields' portion you find all of the derived fields defined for that entity type.

```
Entity Type:Form Number:   100:0

Entity Description:   Circular Arc

Entity Fields:

      subent   igesde

      subent   props
```

```

subent    assocs

double    zt          "ZT Displacement"

double    xyloc []    "Center Point"

double    xystart []  "Start Point"

double    xyend []    "End Point"

```

Derived Fields:

```

double    drv_origin []

double    drv_start []

double    drv_end []

double    drv_radius

double    drv_incangle

double    drv_angs

double    drv_ange

double    drv_chord

double    drv_alength

char      drv_isclosed

```

You may need to reference the DDF entry for many IGES entities. The *list ddf type* command{XE "DDF:*list ddf type* command"} lists this information for entities based on their IGES type (and optionally form) number. Notice that in the above DDF listing for a Circular Arc, the data types of the DE data, the property data, and the associativity data are subentities (subent). They are called subentities because they exist for each entity in the IGES specification. The output of the *list ddf type* command will always contain the subents igesde, props, and assocs. The DDF listing for the subentities are accessed via the *list ddf subent* command{XE "DDF:*list ddf subent* command"}. Refer to the *IGES/Works Command Reference Manual* for additional capabilities of this command.

### **Referencing an Entity's PD Data**

You may reference any field of an entity's Parameter Data (PD). You may want to test a field's value against some constant for the purpose of finding a specific entity or set of entities. Additionally, you may want to extract the field value of a known entity in an IGES file. The syntax for referencing PD data is explained below.

The general syntax for field access is listed below.

```
entity_id.field_name/entity_id.field_name
```

There are two methods for specifying an ID:

A Modify PD Fields dialog shows a list box containing all the PD fields for the entity selected for modification. If there are too many fields for the size of the list box, a vertical scroll bar appears in the list box, with which all the PD fields can be seen. Any PD field can be selected for modification by double clicking on that PD field. After double clicking, the field name and its value appear in the edit box, and can be modified. Also the next and previous show the next or previous field relative to the current field selected.

The second - 'entity\_id' is specified by one of the following five forms:

- a pound sign (#) followed by the entity's DE number (#15).
- a specific type and form number combination enclosed in greater and less than signs (<212:0 >).
- a specific type number reference with a wild card{XE "Wild cards:referencing an entity's PD data"} for the form number (<212:\*>).
- a specific form number reference with a wild card{XE "Wild cards:referencing an entity's form number"} for the type number (<\*:0>).
- a generic reference to all entities (<\*:\*>).

The 'field\_name' in the above general syntax is an entity's field name from the entity's DDF listing. There are three special symbols used in conjunction with the general syntax described above. These symbols are listed below:

- a single dot (.) This symbol always precedes the 'field\_name'.
- a slash (/) This symbol denotes that you are stepping down through the entity hierarchy.
- a double dot and slash (../) This symbol denotes that you are stepping up through the entity hierarchy.

In the following example, you know that the entity with a DE number of 57 is a Circular Arc and you want to extract its start point information. From the above DDF listing for a Circular Arc, you know that the start point data is referenced by the name 'xystart'. Using the special pound sign notation (#) to reference a specific entity, this entity's start point is accessed as follows:

```
#57.xystart
```

On a more global scale, you can use the angle brackets < > and an entity's IGES type and form numbers to reference a geometry field for all entities of a specific type. For example, you may want to access the center point for all Circular Arc entities in the current model. You know that the Circular Arc's IGES entity type number is 100, its IGES entity form number is 0, and the name of the center point field name is 'xyloc'. Combining these pieces of information, the following syntax will reference this desired data:

```
<100:0>.xyloc
```

The following command will find all Circular Arc entities in the Current Selection List that have a 'ZT' displacement equal to 1.4.

```
find entity selection_spec (<100:0>.zt,eq,1.4)
```

Notice in each of the above examples, a period (.) is used between the entity reference (#57) and the field name or between the entity specification (<100:0>) and the field name. This is true for all field name references.

Data fields can be arrays. These arrays could be pointer arrays (as in a subfigure definition), a real array (as in a NURBS surface) or even an array of structures (as in a text entity). References to arrays allow some special syntax which is worth some special discussion.

Array members are enclosed in brackets [ ], and references begin at 0. For example, to load the second knot value in the first knot vector of a NURBS surface (type 128) from your Current Selection List into the variable "my-knot," you could use the following syntax:

```
Get Field Field_Spec <128:*>.knots1[1] Variable my_knot
```

Not only can you select a particular index from an array, you can also select an range of index values. Here are some examples:

```
change selection field_spec <308:*>.des[2:30]
```

```
change selection field_spec <308:*>.des[30:*
```

```
change selection field_spec <308:*>.des[*]
```

The example above

```
change selection field_spec <308:*>.des[*]
```

is the same as:

```
change selection field_spec <308:*>.des
```

which is the same as:

```
change selection field_spec <308:*>.des[0:*
```

If the range extends beyond the end of the array, only the maximum index value allowed will be selected:

```
change selection field_spec <308:*>.des[350:10000]
```

### ***Referencing an Entity's DE Data***

An entity's DE data, due to the wealth of information that it contains, is accessed often within IGES/Works. Therefore, it is important to understand the fields in the DE data and how to reference it. Each IGES entity references its DE data through a subentity called 'igesde'. The DE data, along with the property and associativity data, are classified as subentities because they exist for all IGES entities.

The following command results in a DDF listing of the 'igesde' SubEntity:

```
list ddf subent igesde
```

Here is what it would output:

```
Entity Description:  DE Attributes
Entity Fields:
int      denum      "DE number"
int      type       "DE entity type"
int      form       "DE form"
union    structure  "Structure"
int      version    "Version Number"
ptr      sdptr      "Structure Definition Pointer"
union    font       "Line Font"
char     fnum       "Line Font Number"
ptr      fdptr      "Line Font Definition Pointer"
union    level      "Level"
int      lnum       "Level Number"
ptr      ldptr      "Level Definition Pointer"
ptr      view       "View Pointer"
```

ptr	matrix	"Matrix Pointer"
ptr	ldisp	"Label Display Assoc. Pointer"
int	weight	"Line Weight"
int	subscript	"Subscript"
union	color	"Color"
char	cnum	"Color Number"
ptr	cdptr	"Color Definition Pointer"
fstring	label	"Entity Label"
char	blanking	"Blank Status"
char	usage	"Usage"
char	hierarchy	"Hierarchy"
char	subord	"Subordination"

Most of the fields are defined as single value integers. An example of this is the entity type number (field name 'type') and the entity's line weight (field name 'weight'). The following statements access the 'type' and 'weight' fields for entity numbered DE 33.

```
#33.igesde.type
#33.igesde.weight
```

Some DE fields are more complex such as the color{XE "Color:DE fields"} and line font. This is due to the fact that these fields can have an explicit integer value (representing a specific color or line font value) or a pointer to a definition entity (such as a color definition entity or a line font definition entity). These special cases are identified by the data type 'union'. For example, the line font number and the pointer to the color definition entity of entity numbered DE 17 are referenced below.

```
#17.igesde.font.fnum
#17.igesde.color.cdptr
```

As an example, the following command will place the Level Definition pointer entities of all Parametric Spline Curve entities (type 112) in the Current Selection List.

```
change selection field_spec <112:0>.igesde.level.ldptr
```

**Note:**

*Any reference to a union will be resolved as the first member of the union.*

As an example:

```
igesde.color becomes igesde.color.cnum
```

See Chapter 10, "Working with IGES/Works variable," for a description of the variable types.

## ***Referencing an Entity's Property Pointers***

As with the DE data and the associativity pointers, the property pointers are classified as subentities because each of these three pieces of information are contained in all IGES entities. The following command results in the DDF Listing of the Property SubEntity:

```
list ddf subent props

Here is what it would output:

Entity Description:  PROPERTY POINTERS

Entity Fields:

count      npps          "Number of Property Pointers"

ptr        propdes[]   "Property Pointers"
```

The two fields of the property subentity are the number of pointers to property entities and an array containing the pointers (DE numbers) to the property entities. The following syntax refers to the number of property pointers of entity numbered DE 49 in the current model.

```
#49.props.npps
```

The syntax below refers to the third property pointer of entity numbered DE 77.

```
#77.props.propdes[2]
```

## ***Referencing an Entity's Associativity Pointers***

As with the DE data and the property pointers, the associativity pointers are classified as subentities because this information is contained in all IGES entities.

The following command results in the listing of the Associativity SubEntity:

```
list ddf subent assocs
```

Here is what it would output:

```
Entity Description:  ASSOCIATIVITY POINTERS

Entity Fields:

count      nbps          "Number of Backpointers"

ptr        backdes[]   "Backpointers"
```

The two fields of the associativity subentity are the number of pointers to associativity entities and an array containing the pointers (DE numbers) of the associativity entities. The following syntax refers to the number of associativity pointers of entity numbered DE 219 in the current model.

```
#219.assocs.nbps
```

The sample below refers to the first associativity pointer of entity numbered DE 93.

```
#93.assocs.backdes[0]
```

## ***Referencing an Entity's Derived Fields***

Derived fields are entity fields that are calculated based on the fields defined in the IGES specification. These fields are not defined in IGES but are available to you through the DDF{XE "DDF:referencing an entity's derived fields"}. You may access and use derived fields just like other entity fields. Examples of derived fields are:

- radius of circles
- length of lines
- area of surfaces

These fields provide unique ways of specifying entities. As an example, the following command finds independent Circular Arc entities with a radius of less than 2.3 in the Current Selection List.

```
find entity selection_spec (<100:0>.drv_radius, lt, 2.3)
```

When querying the DDF of an entity, its derived fields are listed as well. From your perspective, there is no difference between IGES entity fields and derived fields.

## Chapter 10. Navigating the IGES Entity Hierarchy

*In this chapter you learn about using the*

- change selection command
- find entity command

Before you begin you should

- have a working knowledge of the IGES/Works Data Definition File. See Chapter 8, "Understanding the Data Definition File," for information on this.

### ***Using the Change Selection Command***

The *change selection* command allows you to navigate the entity hierarchy structure of an IGES file. It generates a hierarchical selection list of entities in the current file based upon the path you tell the software to traverse.

A selection list created by this command contains entities in a specific level within an entity hierarchy (tree structure). This command has many similarities to the *change directory* command in the UNIX operating system. As a result, you can move up and down the tree structure using the special symbols double dot (..) and the slash (/). The double dot indicates that you want to go to the next higher level in the entity hierarchy and the slash indicates that you want to go down one level deeper in the entity hierarchy.

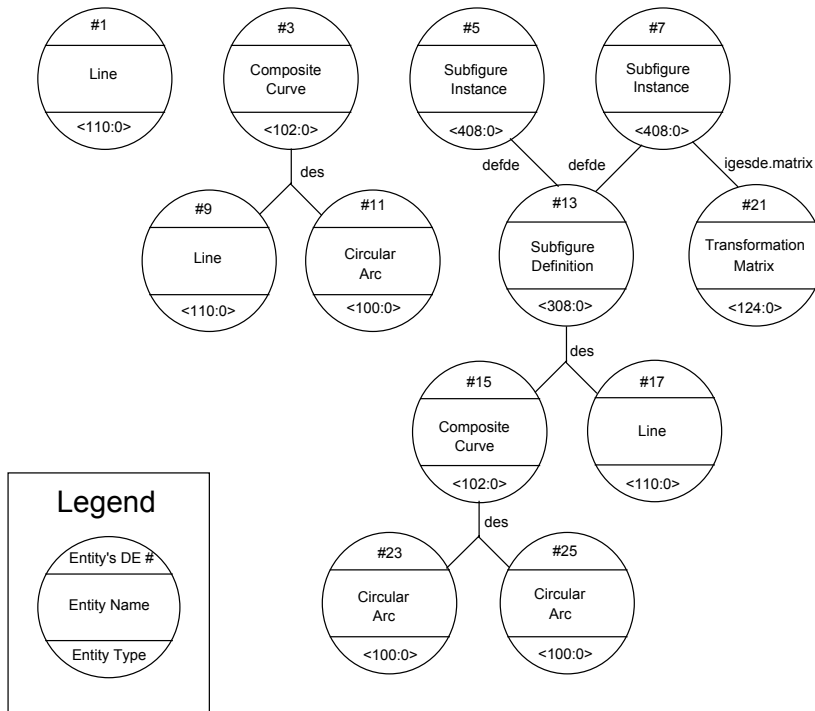
#### **Note:**

*You need a working knowledge of the IGES/Works Data Definition File (DDF{XE "DDF:using the change selection command"}) to be able to use this command. Refer to "The Data Definition File representation of the IGES specification," in Chapter 9, "Understanding the Data Definition File," for a description of the DDF.*

Using the entity type and form pairs (such as <100:0>) and wild card{XE "Wild cards:using with the change selection command"} characters (such as <308:\*> and <\*. \*>) causes the *change selection* command to iteratively traverse each branch of the entity structures. Consequently, non hierarchical lists are generated in these cases.

The *change selection* command takes its input as either the current file or the Current Selection List and places all entities found with this command in the Current Selection List. You may specify either an absolute location or a relative location to traverse to a particular point in the entity hierarchy.

Lastly, only DDF fields which are references to other entities are allowed in the field specification of this command. We shall use the Sample Entity Hierarchy shown on the next page to show how this command works.



**Sample Entity Hierarchy Diagram**

The format of the *change selection* command is given below.

```
change selection field_spec user-defined-spec
```

The user-defined-spec may take many forms and it is used to navigate the entity hierarchy. As an example, the following command will place in the Current Selection List all independent entities in the current file.

```
change selection field_spec /
```

In this example, the slash (/) by itself instructs the software to go to the root or highest level within the current file (go to the independent entities) to perform its search. The Current Selection List will be loaded with four entities; DE numbers 1, 3, 5, and 7.

The following command will locate all independent Subfigure Instance entities (type 408, form 0). As a result, entities DE 5 and 7 are placed in the Current Selection List.

```
change selection field_spec /<408:0>
```

Suppose you know that the entity with a DE value of 3 is an independent Composite Curve (type 102, form 0) and you want to know what entities it references (the child entities of the Composite Curve DE 3). The following command places entities DE 9 and 11 in the Current Selection List.

```
change selection field_spec /#3.des/<*:*>
```

If entity DE 3 was already in the Current Selection List rather than the current file, then the following command would retrieve entities that it references. The difference between the above command and the following one is that the entity specification is not prefaced with a slash. The slash in the above example indicates that the entity is an independent entity in the current file whereas in the example below the lack of a slash indicates that the input source is the Current Selection List.

```
change selection field_spec #3.des/<*:*>
```

Let us examine the above field specification. You are instructing the software to go to entity DE 3 which has a DDF field named 'des.' You can verify this field name by entering the following command{XE "DDF: using the *list ddf* command"}:

```
list ddf type 102
```

You can further instruct the software to navigate one level deeper (as indicated by the slash) and retrieve all entities (denoted by the <\*:\*>) referenced in the 'des' field. As a result, the entities DE 9 and DE 11 are placed in the Current Selection List.

Notice that in the diagram there are two occurrences of a Composite Curve entity (type 102). There are two unique paths to traverse to access each of these entities. For the Composite Curve with a DE value of 3, it is referenced by the following command.

```
change selection field_spec /<102:0>
```

However, to access the other Composite Curve with a DE value of 15, you need to step through the entity hierarchy. Since there are two different paths to this Composite Curve there are two forms of the *change selection* command to reference this entity.

```
change selection field_spec /#5.defde/<308:0>.des/<102:0>
```

```
change selection field_spec /#7.defde/<308:0>.des/<102:0>
```

The above commands results in the entity with a DE value of 15 being placed in the Current Selection List. The differences in these last two commands is that the inherited parent data will differ for the Composite Curve, because of the different parent entities.

In a slightly different form, the command below places the Composite Curve (DE 15) in the Current Selection List twice. The double occurrence is due to the generic reference to the Subfigure Instance (<408:0>). The command finds two Subfigure Instance entities at the independent level and therefore finds the same entity from traversing two different branches in the file (branch one is DE 5-13-15 and branch two is DE 7-13-15).

```
change selection \  
field_spec /<408:0>.defde/<308:0>.des/<102:0>
```

In all of the above examples, the entities have been accessed by an absolute reference. You can also access entities by relative references. Assuming that the above command was last executed, you could transfer to the Line entity (DE 17) with the following command.

```
change selection field_spec ../<308:0>.des/<110:0>
```

In the above command, the Current Selection List is the input source and the double dot (..) informs the software to move up to the next higher level in the entity hierarchy which is the Subfigure Definition. The rest of the field specification traverses back down the hierarchy to the Line entity which will be placed in the Current Selection List.

### **Using the Find Entity Command**

The *find entity* command navigates the entity hierarchy structure of an IGES file and generates a hierarchical selection list of entities in the current file based upon the entity selection criteria you defined. A selection list created by this command contains entities in a specific level within an entity hierarchy (tree structure).

This command has many similarities to the change directory command in the UNIX operating system. As a result, you may move up and down the tree structure using the special symbols double dot (..) and the slash (/). The double dot indicates that you want to go to the next higher level in the entity hierarchy and the slash indicates that you want to go down one level deeper in the entity hierarchy.

**Note:**

*You need a working knowledge of the IGES/Works Data Definition File (DDF) to be able to use this command{XE "DDF:using the find entity command"}. Refer to "The Data Definition File representation of the IGES specification," in Chapter 8, "Understanding the Data Definition File," for a description of the DDF.*

There are many similarities between the *find entity* command and the *change selection* command. Both the current file and the Current Selection list may be input sources. All entities found by these commands are placed in the Current Selection List. Whereas the *change selection* command only accepts field names which are references to other entities, the *find entity* command accepts any of the field names defined for the entity in the DDF. As a result, the *find entity* command can locate entities based on any combination of DE and PD data.

The general format of this command is shown in the following example.

```
find entity field_spec user-defined-value \  
  
selection_spec user-defined-criteria
```

The 'field\_spec' modifier is optional. If omitted, then the input source is by default the Current Selection List. Otherwise depending on the value of the 'field\_spec', the input source may be either the current file or the Current Selection List. The 'selection\_spec' modifier defines specifically the attribute values of those entities that are to be located.

As an example the following command finds all Parametric Spline Curve entities (type 112) in the current file.

```
find entity field_spec / selection_spec (igesde.type,eq,112) \  
  
-recursive
```

In this example, the 'field\_spec' is defined as a slash which means to search all independent entities in the file. However, the addition of the 'recursive' switch indicates that not only the independent entities are to be searched but all entities in the file regardless of their subordination status. The user criteria in the 'selection\_spec' states that the type field in the entities' DE field should be compared to the value 112. All matching entities are placed in the Current Selection List.

In the next example all Composite Curve entities (type 102) in the current file that point to a Line entity (type 110) are placed in the Current Selection List.

```
find entity field_spec / \  
  
selection_spec (<102:*>.des/<*. *>.igesde.type,eq,110) \  
  
-recursive
```

The combination of a slash for the field specification and the 'recursive' switch indicates that the input source is all entities in the current file. The selection criteria indicates that the DE type field (igesde.type) of each child entity (.des/) of every Composite Curve (<102:\*>) should be compared for equality to the number 110 (a Line entity). All child entities satisfying the criteria are placed in the Current Selection List.

In the following example, it is known that entity DE 5 is a Subfigure Instance. The command will find all children of this Subfigure Instance that have a property pointer.

```
find entity field_spec /#5.defde/<308:*>.des \  
  
-recursive
```

```
selection_spec (props.npps,gt,0)
```

The input source for the above command is all child entities (.des) of the Subfigure Definition entity (<308:\*>) that is instanced by entity DE 5 (#5.defde). The number of property pointers for each of these entities (props.npps) is examined. Values greater than zero indicate a child entity that references property entities. These child entities are placed in the Current Selection List. As a final example, all Circular Arc entities (type 100) in the Current Selection List that have a derived origin with an X coordinate value less than 2.0 are placed in the Current Selection List.

```
find entity selection_spec (<100:0*>.drv_origin[0], lt, 2.0)
```

Notice that in this example, there is no 'field\_spec' modifier. Therefore, the input source is the entities in the Current Selection List. If there are any Circular Arc entities (<100:\*>) in the list, the X coordinate of their derived origin point (drv\_origin[0]) is compared (lt) to 2.0. Upon completion of this command, the list contains only those entities that meet the user criteria that was specified. Notice also that the field name in the above command is 'drv\_origin' which does not match any PD field in the IGES specification. This is an example of a derived field, calculated by IGES/Works based on the IGES data. Refer to "Referencing an entity's derived fields," in Chapter 9, "Understanding the Data Definition File," for a description of this feature.

## Chapter 11. Working with IGES/Works Variables

*In this chapter you learn about*

- IGES/Works variables
- creating IGES/Works variables
- IGES/Works system variables
- using IGES/Works variables in commands
- IGES/Works functions with variables
- using IGES/Works functions with variables

### **About IGES/Works variables**

IGES/Works variables{XE "Variables"} are similar to variables in any programming language. Variables store data values. They are used in scripts and as the value in a command modifier's name/value pair. Within scripts they serve a variety of purposes but are especially useful when creating entities, generating reports{XE "Reports:using variables in scripts to generate"}, and parsing data from an ASCII file into a script.

### **Creating IGES/Works variables**

IGES/Works variables have three characteristics:

- name
- data type
- scope

The variable name is the symbolic name assigned to the variable. If the variable is an array, the array dimensions are specified with the name. Array indices begin with zero (0) and go to one less than the maximum dimension. For example, the elements of a three element array named ABC are referenced by ABC[0], ABC[1], and ABC[2].

There are three data types:

- integer
- floating point
- fixed length character string

The default length of a fixed length character string is 1024 characters if there is no explicit dimension. The maximum string length is also 1024 characters.

The variable's scope specifies the level of accessibility of the variable within IGES/Works. If the scope is local, the variable can only be used in the script defining the variable or on the command line if the variable is defined on the command line. If the scope is global, then the variable can be used in all scripts and on the command line.

The *define variable* command{XE "Variables:using the define variable command"} creates variables. The command requires you to enter the name and optionally the data type (default is integer) and the scope (default is local). Below are several examples of how to use this command. Refer to the *IGES/Works Command Reference Manual* for additional information on this command.

The following command creates a variable named 'count' that is an integer and has a local scope.

```
define variable name count
```

The following command creates a variable named 'points' which is a five element array of floating point numbers with a global scope.

```
define variable name points[5] -float -global
```

The following command creates a 12 element array of character strings named 'lines' in which each string is 80 characters long and has a local scope. Note that the first array dimension specifies the number of elements and the second array dimension specifies the string length.

```
define variable name lines[12][80] -fstring
```

The following command creates an integer array named 'circle' whose dimension is specified by the variable 'count'. The dollar sign symbol (\$) before the variable, 'count' indicates to use the value of the variable.

```
define variable name circle[$count] -integer
```

### **About IGES/Works System Variables**

IGES/Works defines and maintains several system variables{XE "Variables:system"} which can be accessed both on the command line and within scripts. These variables represent the current state of the software and useful pieces of information. They are described below. The values of these variables are automatically maintained and updated by IGES/Works.

csi_count	The number of entities in the Current Selection List.
cur_model	The name of the current model.
num_models	The number of models currently allocated.
cur_nest	Since scripts may call other scripts, you may want to know how many levels deep you currently are. This variable indicates the current level of nesting within scripts. Cur_nest is set to 1 at the first script level.
cur_wp	The current working path. Within a model there may be many nested levels of parent/child relationships (an entity is dependent on another entity which is dependent on another entity etc.). This variable keeps track of where in the entity hierarchy you currently are.
date	Today's date.

error_return	The error value returned from the last command.
float_null	Represents a null floating point number. Used in conditional statements and other comparisons.
fstring_null	Represents a null fixed length string. Used in conditional statements and other comparisons.
integer_null	Represents a null integer number. Used in conditional statements and other comparisons.
argc	Represents the number of arguments passed into IGES/Works from the command line.
argv	Represents an array of arguments passed into IGES/Works from the command line.
graphics_on	Indicates whether the current session is in graphics mode (set to 1) or not (set to 0)..

### **Using IGES/Works Variables in Commands**

Variables may be used in IGES/Works commands{XE "Variables:using in commands"} in place of explicit data values. This can be especially useful to represent long pieces of data that you may not want to type each time you need to refer to the data value. String variables can be concatenated to form a single value. In a script, you could prompt the user for the name of a file to read and use both a directory variable and a file name variable. Examples are listed below.

```
read iges model test file \
    /users/john/project/igesfiles/test.igs
read iges model test file $dir/test.igs
read iges model test file $dir/$name
```

When using variables as the value of a command modifier, the variable name must be preceded by a dollar sign (\$). When a command is explicitly expecting the name of a variable, the variable should NOT be preceded by a dollar sign.

### **About IGES/Works Functions with Variables**

IGES/Works contains a complete set of functions that can be applied to variables{XE "Variables:using with functions"}. These functions include mathematical operations, string manipulations, matrix related calculations, and other functions. The various functions and their names are listed below.

## Math Functions

ADDF	Adds two floating point numbers.
ADDI	Adds two integer numbers.
ASSIGNF	Assigns a value to a floating point variable.
ASSIGNI	Assigns a value to an integer variable.
DEGREES	Converts radians to degrees.
DISTANCE	Evaluates the vector distance between two 3D points.
DIVF	Divides two floating point numbers.
DIVI	Divides two integers.
EXPONENT	Raises a number to a power.
FLOAT	Converts an integer to a floating point number.
INT	Truncates a floating point to an integer variable.
MULTF	Multiplies two floating point numbers.
MULTI	Multiplies two integers.
NINT	Rounds a floating point number to an integer and assigns the value to an integer variable.
RADIANS	Converts degrees to radians.
SQRT	Evaluates the square root of a floating point number.
SUBF	Subtracts two floating point numbers.
SUBI	Subtracts two integers.

**String Manipulation and Conversion***{XE "Conversion:string manipulation and conversion functions"}* **Functions**

ASSIGNST	Assigns a value to a string variable.
CATST	Concatenates two strings.
FMTFLT	Formats a floating point number into a string variable.
FMTINT	Formats an integer into a string variable.
LOWER	Converts all characters in a string to lowercase characters.
STLEN	Evaluates the length of a string.
STRFLT	Formats a string into a floating point variable.
STRINT	Formats a string into an integer variable.
UPPER	Converts all characters in a string to uppercase characters.

**Matrix Functions**

GEN_TM	Generates a 3x4 IGES Transformation Matrix from an X,Y,Z angle and X,Y,Z offset.
GET_TM_ANGLES	Extracts the angles (in radians) from a 3X4 transformation matrix.
GET_TM_SCALE	Extracts the scale from a 3X4 transformation matrix.
TRANSFORM	Applies a 3X4 IGES Transformation Matrix to a 3D point.
TRANSPOSE	Transposes the rotation portion of a 3x4 IGES Transformation Matrix.

## Other Functions

EXISTS	Check if a file exists.
GET_EXT	Extracts all characters after the rightmost period (".") in a string.  This is useful when you want to check a file's extension.
GET_ROOT	Extracts all characters up to the last occurrence of a period in a string. This is useful when you want to replace the file's extension.
GET_TAIL	Extracts all characters after the last occurrence of a slash ("/") in a string and before the period (".") in a string. This is useful in extracting a file's basename.
MAKE_UNIQUE_NAME	The purpose of this function is to create a unique name. This is useful for generating file names, model names, and other situations where identical names would be a problem. Names are made unique by adding a suffix composed of an underscore (_) and a number (for example '_1', '_2', '_3'). This routine inputs a string and checks for a numerical suffix. If the suffix exists, the number is extracted from the string, incremented by one, and a new string created. If no suffix exists, '_1' is added to the end of the string.
READ	Checks if the user has read permission on a file.
READWRITE	Checks if the user has both read and write permissions on a file.
WRITE	Checks if the user has write permission on a file.

## Using IGES/Works Functions with Variables

All IGES/Works functions are executed using the *evaluate function* command. Within this command you specify the name of the function, the output variable{XE "Variables:using with functions"}, and one or two (depending on the command) input arguments. The general syntax{XE "Command:syntax:evaluate function command"} is listed below. Refer to the *IGES/Works Command Reference Manual* for additional information on this command.

```
evaluate function args (function_name, output_var, \  
    input_value_1, [input_value_2])
```

Several examples of using functions are presented below. Note that the output variable is expecting the name of a variable and therefore is NOT preceded by a dollar sign whereas the input arguments are expecting a value and variables are preceded by a dollar sign. Assigning a value to a variable (xval = 5.3).

```
evaluate function args (assignf, x_val, 5.3)
```

Adding two integer numbers (result = y\_val + delta).

```
evaluate function args (addi, result, $y_val, $delta)
```

Concatenating two strings (file = name || ".igs").

```
evaluate function args (catst, file, $name, ".igs")
```

Applying a transformation matrix to a 3-D point.

```
evaluate function \  
    args (transform, new_pt, $old_pt, $matrix)
```

## Chapter 12. Writing Scripts

*In this chapter you learn*

- about writing and modifying scripts

Before you begin you should be familiar with the contents of the following chapters:

- Chapter 8. "Entering IGES/Works commands."
- Chapter 9. "Understanding the Data Definition File."
- Chapter 10. "Navigating the IGES entity hierarchy."
- Chapter 11. "Working with IGES/Works variables."

### **Executing Python Scripts**

You can execute a Python script from within IGES/Works. A menu option "Execute Python Scripts" is provided under the "Scripts" menu. Python is a standard, easy to learn, and popular scripting language. This also allows IGES/Works programmers to use a rich set of functionality provided by available Python libraries. For example, creating custom GUI, running the scripts from a web application, or writing and using advanced mathematics functions in the scripts. For more information, refer to the "IGES/Works Python Scripting Reference Manual".

You can also use standard, IGES/Works scripting language to write scripts and macros. This scripting language is described in this chapter.

### **Writing and Modifying Scripts**

You can write your own scripts or modify existing ones by using any text editor. Scripts can be as simple as four lines that do something simple (like reporting{XE "Reports:using scripts to generate"} the number of points found on a layer) or can be very long and complex (like a whole series of flavorings that modify, copy{XE "Copying entities:using scripts"}, delete{XE "Deleting entities:using scripts"}, and convert{XE "Converting entities:using scripts"} entities to make an IGES file adhere to standards). To make them available, place them in the standard IGES/Works directory named scripts. You can also assign them to the Custom menus{XE "Custom menu:assigning scripts"}.

To write scripts, you need to be fully conversant with IGES/Works commands and command structure (in general).

You should also be aware of

- the *Find Entity* and *Change Selection* commands (in particular)
- IGES entity structures
- simple programming constructs

To assist your script writing, you should become familiar with

- command journaling
- GUI recorder
- switches of the *Execute Script* command for debugging.

### **Command Journaling**

All commands entered on the command line are automatically captured in a journal file. By default, the journal file is "igesworks.jrn" created in the directory where IGES/Works was started up.

## GUI recording

Most of the actions you can perform using the buttons and menus are also, by default, captured in the journal file as IGES/Works commands. There is also a Recorder Dialog, available from the Options menu of the Main Window, which controls whether GUI actions are recorded. The dialog also allows you to change the path and name of the journal file.

## Switches of the Execute Script Command for Debugging

Echo_P	will echo each line of the script as it is being parsed (read and interpreted). Note, if calling a "sub"script from another script, the switch should be used on the "sub"script call also.
Echo_X	will echo each line of the script as it is being run.
Step	allows manual stepping through each line of the script as it executes.
Stop	stops the script when an error occurs.

Writing and modifying a script involves using IGES/Works commands using a framework that consists of the following elements:

- passing data into the script
- using conditional statements
- using loop structures
- issuing messages from the script.

## Passing Data into the Script

There are two main methods you can use to pass data into a script:

- by prompting the user to enter data while the script is running
- by using the *args* modifier of the *execute script* command to pass data from one script to another, or when running a script from the command line.

**Prompting the user to enter data while the script is running** is the most common method of passing data into a script. This allows users to query IGES files without having to know anything about IGES.

IGES/Works provides the following prompt commands:

- get data
- get filename
- get menu
- get dialog

## Using the Get Data Command

Use the *get data* command to get interactive input by creating a dialog{XE "Dialog:using the *get data* command"}.

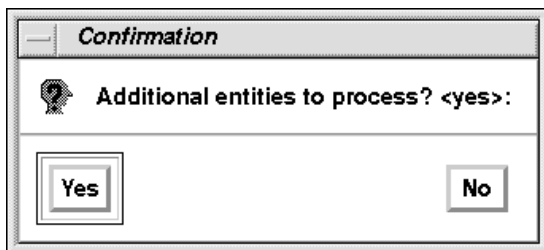
**For example**, the following script creates a data dialog:

```
# get_data.scr

# Run this script to create the 'get data'
dialog define variable name answer -fstring

get data args (answer, "Additional entities to process?") -yes
```

displays the following prompt:



## Using the Get Filename Command

Use the *get filename* command to create a file dialog{XE "Dialog:using the *get filename* command"} in which the user is prompted to enter the name of a file.

**For example**, the following script creates a filename dialog:

```
# get_filename.scr

# Run this script to create the 'get filename' dialog

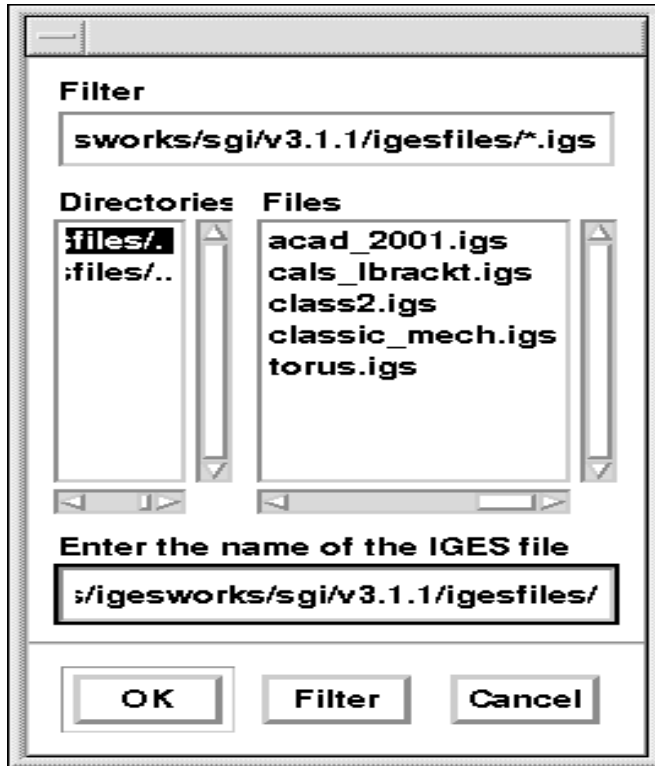
define variable name my_var -fstring

get filename variable my_var \

    prompt (Enter the name of the IGES file) \

    directory (/users/igesworks/v2.0.0/igesfiles) \

    extension (igs)
```

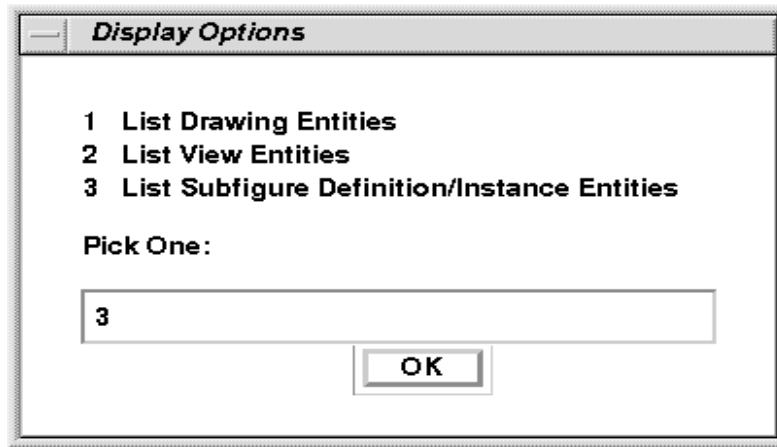


### Using the Get Menu Command

Use the *get menu* command to construct a menu dialog{XE "Dialog:using the *get menu* command"}. Many times in scripts a series of *echo data* commands are used with a *get data* command to present the user with a “menu” of options to choose from. A simpler and cleaner method with the GUI is to use the new *get menu* command which allows you to construct a “menu” dialog.

**For example,** the following script creates a menu dialog:

```
# get_menu.scr
#
# Run this script to create the 'get menu' dialog
define variable name choice -integer
get menu args (1, List Drawing Entities) \
    args (2, List View Entities) \
    args (3, List Subfigure Definition/Instance Entities) \
    title (Display Options) \
    prompt (Pick One:) \
    default 3 \
    variable choice
```



### Using the Get Dialog Command

Use the *get dialog* command to construct a more complicated dialog. Controls include check boxes, list boxes and text entry fields.

**For example**, the following script creates a dialog:

```
# get_dialog.scr

#

# Run this script to create a dialog.

define variable name okcancel -integer

define variable name check1 -integer

define variable name check2 -integer

define variable name check3 -integer

get dialog title (Query IGES File) \

  left 20 top 20 width 200 height 250 \

  out_info okcancel num_controls 3 \

  control (checkbox, 1, "Query Drawings & Views", \

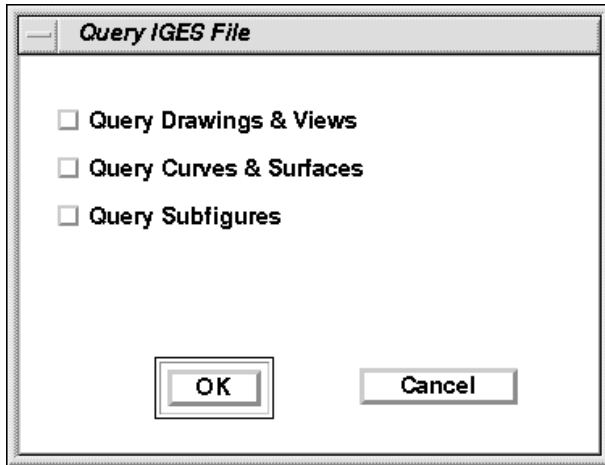
    check1, 0, 20, 30, 200, 22, 0) \

  control (checkbox, 2, "Query Curves & Surfaces", \

    check2, 0, 20, 60, 200, 22, 0) \

  control (checkbox, 3, "Query Subfigures", \

    check3, 0, 20, 90, 200, 22, 0)
```



**Using the *args* modifier of the *execute script* command**, together with the *get argument* command and the *put argument* command, you can pass data into and out of a script – either directly from the command line or from another script.

## Using Conditional Statements

You can make the performance of IGES/Works commands conditional in scripts by using an *if* statement. This statement tests a condition and, depending on the result of the test, will execute or skip one or more commands.

The *if* statement works with the following related commands:

- *if*
- *else*
- *else\_if*
- *end\_if*

**The basic syntax of the *if* command** {XE "Command:syntax:of the *if* command"} requires the two values to be compared and the comparison operator to be present. For example:

```
if condition ($color_num, eq, 5)

    echo data args (The color is red)

end_if
```

**Where:** `$color_num` is the variable value, `eq` is the comparison operator, and `5` is the fixed value with which the variable value is to be compared.

The standard comparison operators are:

- • `eq` (equal to)
- • `ne` (not equal to)
- • `lt` (less than)
- • `le` (less than or equal to)
- • `gt` (greater than)
- • `ge` (greater than or equal to)

The following special comparison operator is also available:

- pm (pattern matching)

The pattern matching operator allows you to match specific patterns in string variables only. For example, as shown in the sample below, you could use it to check that a part number conforms to your company standard (which specifies that a valid part number is composed of three numeric characters, a dash, and four alphabetic characters).

```
if condition ("nnn-aaaa", pm, $part)
.
.
.
end_if
```

**Where:** the letters “n” and “a” represent special wild card characters. IGES/Works contains a set of six wild card characters. Refer to “Using and modifying wild card characters” in Chapter 8 “Entering IGES/Works commands” for more information about this.

**Note:**

*The value containing the wild card characters MUST be the first value in the conditional statement.*

**Use the *else* and *else\_if* commands** to increase flexibility in conditional statements. The *else\_if* command operates in exactly the same way as the *if* command (it allows you to add another condition to the *if* condition), except that it must appear between an *if* and an *end\_if* command. The following two examples illustrate the *else* and *else\_if* command syntax{XE "Command:syntax:else and else\_if"}:

```
if condition ("aaa", pm, $color_name)
    echo data args (The color is red.)
else
    echo data args (The color is blue.)
end_if
```

In the next example, the value can be any integer value, but the values 0-5 are acceptable.

```
if condition ($value, gt, 5)
    echo data args (Value too large.)
else_if condition ($value, lt, 0)
    echo data args (Value too small)
else
    echo data args (Valid value provided.)
end_if
```

## Using Loop Structures

IGES/Works scripts offer three kinds of loop structure:

- *for*
- *foreach\_entity*
- *while*

**Use the *for* loop** to execute one or more commands a fixed number of times. The *for* loop uses the following commands:

- *for* opens the loop and requires as inputs an index variable, a terminate value, an optional initial value for the index variable, and an optional increment value.
- *break\_for* causes execution to exit the current loop and resume with the first command after the matching *end\_for* command.
- *continue\_for* causes execution to jump to the matching *end\_for* statement and perform the next pass through the current *for* loop.
- *end\_for* closes the *for* loop.

In the following example, the variable named “ndx” is incremented by 2 each time through the loop until ndx is equal to or greater than 75.

```
for variable ndx terminate 75 step 2
.
.
end_for
```

**Use the *foreach\_entity* loop** to perform commands on all entities in the Current Selection List. The *foreach\_entity* loop uses the following commands:

- *foreach\_entity* opens the loop.
- *break\_foreach\_entity* causes execution to exit the current *foreach* loop and resume with the first command after the matching *end\_foreach\_entity* command.
- *continue\_foreach\_entity* causes execution to jump to the matching *end\_foreach\_entity* statement and perform the next pass through the current *foreach\_entity* loop.
- *end\_foreach\_entity* closes the *foreach\_entity* loop.

The following example illustrates the *foreach* loop syntax:

```
foreach_entity
get field field_spec igesde.color.cnum \
    variable color
get field field_spec igesde.type variable type
echo data args (Entity $type_num is color_name[$color])
end_foreach_entity
```

**Use the *while* loop to** execute one or more IGES/Works commands until a specific condition is achieved. The *while* loop uses the following commands:

- *while* opens the loop and requires, as input, the condition to be met.
- *break\_while* causes execution to exit the current while loop and resume with the first command after the matching *end\_while* command.
- *continue\_while* causes execution to jump to the matching *end\_while* statement and perform the next pass through the current *while* loop.
- *end\_while* closes the loop.

The following example illustrates the *while* loop syntax:

```
while condition ($limit, le, 15)
.
.
end_while
```

## Issuing Messages from the Script

You can use the *echo data* command to make scripts issue messages, containing any combination of text and variable data, and display them in the Main window text display area.

The following examples illustrate how to use this command:

```
echo data args (Now performing the entity modification.)

echo data args (There are $csl_count entities in the \
current selection list.)

echo data args (File $fname in directory $dname will \
be read.)
```

You can also use Generic File commands to output highly formatted messages. See the description of “buffers” in Chapter 13, “Working with Generic Files.”

## Chapter 13. Working with Generic Files

*In this chapter you learn about*

- using ASCII files
- generating reports from ASCII files
- reading ASCII files.

### **About using ASCII Files**

IGES/Works allows you to access non-IGES ASCII files through the Generic File Management commands. You can use these commands to generate reports containing data from IGES files formatted to your specification.

Such reports allow you to document the contents of an IGES file or extract pertinent information and create a data file used in the operation of manufacturing equipment. You can also read ASCII files containing raw data from which you can create IGES entities{XE "Creating entities:from raw data in ASCII files"}. You would usually use this technique from within a script.

### **Generating Reports from ASCII Files**

There are several sources of the data that are formatted into a report. The data can come from IGES entity fields, script variables, or explicit data. You can send the report output either to the terminal or to a file.

The seven step procedure for generating reports{XE "Reports: generating from ASCII files"} in IGES/Works is outlined below with the related command for each step.

1. Open the ASCII file for writing (*open file -write*).
2. Define variables which contain data to be formatted and written to the file (*define variable*).
3. Define a buffer and its appropriate formatting parameters used in outputting data to the file (*define buffer*).
4. Format the variable/data into the buffer (*format field*).
5. Write the buffer to the output file (*write record*).
6. Free the buffer when the file is complete (*free buffer*).

Close the ASCII file (*close file*).

The procedure for reading ASCII files is very similar to that for generating reports. The steps are reversed. In the following section, you will find the listing of a script that demonstrates how to work with ASCII files. You can extrapolate the report generation process from this script listing. You can also look at the examples under the *Open File* command in the *IGES/Works Command Reference Manual*.

### **Reading ASCII Files**

The ability to read non-IGES ASCII files is useful for parsing data from a file and loading variables with this data. This helps when you need to generate IGES entities that are defined by large amounts of data. You can also use it in electrical applications – when you need to look up component data and add information to an IGES file to enhance the intelligence of the file.

The process of reading non-IGES ASCII files is very similar to writing ASCII files. The seven step procedure for reading ASCII files in IGES/Works is outlined below, with the related command for each step.

1. Open the ASCII file for reading (*open file -read*).
2. Define variables which contain data to be read and parsed from the file (*define variable*).
3. Define a buffer and its appropriate parsing parameters used in inputting data to the file (*define buffer*).
4. Read a record from the input file and place it in the buffer (*read record*).
5. Parse the buffer data into variables (*parse field*).
6. Free the buffer when you are finished reading the file (*free buffer*).
7. Close the ASCII file (*close file*).

The sample script below reads data from an external ASCII file and creates IGES Line entities{XE "Creating entities:from raw data in ASCII files:a sample script"}. This sample script demonstrates the procedure outlined above and related commands. The sample input ASCII data file that follows it contains a partial listing of the data file. The data file contains the values defining the Line entities to be created. There is no particular format to this raw data file. You can format the data any way you want. The script you write to read the data file will have to know the data format.

### Sample ASCII File Reading Script

```
#####  
  
# This script reads records from the ASCII data file  
  
# 'lines.dat' into a buffer. Data values are extracted  
  
# from the buffer and placed in IGES/Works variables.  
  
# The data values are used to create Line entities  
  
# (type 110). The newly created Line entities are placed  
  
# in a model and displayed after the data file has been  
  
# processed.  
  
#  
  
# Each record in the data file contains the name of the  
  
# entity to create (hence the keyword 'LINE',) the start  
  
# point coordinates (X,Y,Z), the end point coordinates  
  
# (X,Y,Z),the color value, and the line font value
```

```

*****

#

#   Define some variables to store the data values read

#   from the ASCII file.

#

define variable name iti_id

define variable name out_de

define variable name font_num

define variable name color_num

define variable name sval -fstring

define variable name line -fstring

define variable name start[3] -float

define variable name end[3] -float

define variable name error

define variable name index

#

#   Define a buffer to hold a record from the ASCII file.

#

define buffer name sval

#

#   Allocate a model named 'new_lines' to store the entities

#   that are created.

#

allocate model name new_lines -no_autofree -force

#

```

```

#      Open the file 'lines.dat' located in the IGES/Works
#      datafiles directory.
#
open file name datafiles/lines.dat id iti_id
#
#      Loop through the data file, reading records and creating
#      line entities.
#
echo data args (parsing data file...)

while condition (1, eq, 1)
#      Read a record from the data file and place it in
#      the buffer.
      read record id $iti_id buffer sval out_info error
#      Exit this while loop when a read error is
#      encountered.
      if condition ($error, eq, 0)
          break_while
      end_if
#      Parse the first data field in the record into the
#      'line' variable.
      parse field buffer sval field_num 1 variable line
#      If the record contains data for a line entity,
#      process it.
      if condition ($line, eq, LINE)

```

```

#           Extract the start point coordinates into a variable.

parse field buffer sval field_num 2 variable start[0]

parse field buffer sval field_num 3 variable start[1]

parse field buffer sval field_num 4 variable start[2]

#           Extract the end point coordinates into a variable.

parse field buffer sval field_num 5 variable end[0]

parse field buffer sval field_num 6 variable end[1]

parse field buffer sval field_num 7 variable end[2]

#           Extract the entity's color value from the buffer.

parse field buffer sval field_num 8 variable color_num

#           Extract the entity's line font value from the buffer.

parse field buffer sval field_num 9 variable font_num

#           Create the line entity based on the data parsed from

#           the data file's record.

create iges_entity type 110 de_number out_de \

model new_lines pd_fields ($start, $end) \

de_color ($color_num) de_font ($font_num)

#           Keep a running tally of the number of lines created.

evaluate function args (addi, index, $index, 1)

end_if

end_while

#           Print out the number of lines created.

echo data args ($index lines were created.)

#           Graphically display the newly created entities.

display model

```

```
#      Free the buffer.

free buffer name sval

#      Close the ASCII file containing the line data.

close file id $iti_id
```

### **Sample input ASCII Datafile 'lines.dat'**

```
LINE 2.5 2.5 0.0 3.375 2.5 0.0 2 1

LINE 2.5234375D0 2.6875 0.0 3.3984375D0 2.6875 0.0 2 1

LINE 2.5546875D0 2.9375 0.0 3.4296875D0 2.9375 0.0 2 1

LINE 2.53125 2.75 0.0 3.40625 2.75 0.0 2 1

LINE 4.1171875D0 3.1875 0.0 4.3671875D0 3.1875 0.0 2 1

LINE 4.09375 3.0 0.0 4.34375 3.0 0.0 2 1

LINE 4.1484375D0 3.4375 0.0 4.3984375D0 3.4375 0.0 2 1

LINE 4.125 3.25 0.0 4.375 3.25 0.0 2 1

LINE 4.1796875D0 3.6875 0.0 4.4296875D0 3.6875 0.0 2 1

LINE 4.15625 3.5 0.0 4.40625 3.5 0.0 2 1

LINE 4.2109375D0 3.9375 0.0 4.4609375D0 3.9375 0.0 2 1

LINE 4.1875 3.75 0.0 4.4375 3.75 0.0 2 1

LINE 2.7109375D0 4.1875 0.0 3.5859375D0 4.1875 0.0 2 1

LINE 2.6875 4.0 0.0 3.5625 4.0 0.0 2 1

LINE 2.7421875D0 4.4375 0.0 3.6171875D0 4.4375 0.0 2 1

LINE 2.71875 4.25 0.0 3.59375 4.25 0.0 2 1

LINE 2.5 2.5 0.0 2.5234375D0 2.6875 0.0 2 1
```

## Appendix A. Introduction to IGES

### ***What is IGES?***

The Initial Graphics Exchange Specification (IGES) is a neutral file format for exchanging data between dissimilar CAD/CAM/CAE systems. You can translate your work into the IGES format and pass that file to someone whose system is unable to directly translate your system's native file format into its own – but *can* read IGES.

In an *Ideal World* there would be tools that allow you to directly translate files from any source system's format to that of any target system. However, it would take a great deal of effort and money to develop all those specialized translators.

Even though IGES adds a step to the translation process, it becomes the only intermediary format your translator needs to deal with. If you and the people you work with use different CAD systems but can read and write IGES all you need is an IGES translator. Ideally, all the time and money saved by not having to write translators to support all the major CAD packages would be spent on making IGES perfect.

IGES is an effective general solution, but may sometimes break down. Particularly when exchanging data between systems with fundamentally different approaches to representing data. It may also break down when the IGES translators provided with the source and target system choose to support different subsets of the total IGES specification.

The IGES specification defines a structured hierarchical file format and the representation of a variety of entities within the format. The entity set includes

- geometry
- annotation
- structure
- associativity
- property
- attribute entities

This entity set is described below:

<b>Geometry Entities</b>	This entity set represents a set of 3D wireframe, surface, and solid modeling primitives. Typical entities include points, lines, arcs, circles, conics, ruled surfaces, B-spline curves and surfaces, and CSG solids. The geometry entities are used as building blocks to create a meaningful model.
<b>Annotation Entities</b>	This group represents a set of 2D engineering drafting entities. Typical entities include general notes, linear dimensions, feature control symbols and sectioned areas. The annotation entities are used to further describe the design and/or manufacturing characteristics of the geometric model.

<b>Structure Entities</b>	These entities are used to create part hierarchy and to view 3D models. Examples are Subfigure Definitions, Views, and Drawings. A Subfigure Definition is an entity that represents a group of geometric and/or annotation entities. A Subfigure Instance is used to instantiate the Subfigure Definition at multiple locations. A View entity defines a viewing orientation of the model. The View entity can specify a scale, rotation, and/or clipping boundary for the model data. A Drawing entity describes the layout of a drawing sheet by describing the relative locations of View entities and annotation entities. The combination of the Drawing and View structures are used to define the presentation of the geometric model in a 2D coordinate system (paper/screen perspective).
<b>Associativity Entities</b>	These entities are used to logically relate two or more entities. For example, the Network Flow Associativity is used to represent electrical signal connectivity. The Dimensioned Associativity is used to relate model geometry and drawing space annotations.
<b>Property Entities</b>	This group represents a set of entities that are used to assign nongraphic data to base entities. A base entity is an entity that represents a meaningful feature in the native CAD/CAM system. For example, a Subfigure Definition may be used to represent a bolt. In this case a property entity can be used to assign a part number to the bolt.
<b>Attribute Entities</b>	The attribute entities are used to assign nongraphic data in exactly the same manner as a property entities do. However, the attribute entities are implemented in such a way that they are more space efficient.

### ***Why you Should use IGES***

Data sharing between engineering disciplines has come to be a necessity. A number of standard formats exist to make this exchange possible. Some of the most popular formats include:

- Initial Graphics Design Exchange Format (IGES)
- AutoCAD Data Exchange Format (DXF)
- Electronic Design Interchange Format (EDIF)
- Standard for the Exchange of Product Model Data (STEP)

- VDAFS (a German Surface Data Exchange Format)
- SET (a French Data Exchange Standard).

Each of these formats has strengths in the application areas they support, but the IGES application domain has the largest scope. This fact alone probably makes IGES the general CAD data exchange format of choice.

IGES also supports implementor-defined entities and relationships. This makes it possible for two CAD/CAM vendors who want to create a tightly bound translation capability, with enough functional overlap between their systems, to make enhancements to IGES by designing their own extensions.

## Appendix B. Scripts

Attr_Demo.scr	This script looks for the network subfigure instance (type 420) entities in a model and then matches their Primary Reference Designator (PRD) with a reference in the file /usr/apl/igesworks/demo/datafiles/attr_demo.dat. It then creates an attribute definition (type 322 form 1) entity{XE "Creating entities:in Attr_Demo.scr"} based on the data it finds in that file.
Delete_Entity.scr	This script allows you to delete an entity{XE "Deleting entities:in a script by DE number"}, where the selection of the entity is by the DE number. The entity must be independent.
Display_IGES.scr	This script allows you to display IGES data. This data can be either the current model, a specific subfigure definition, a specific network subfigure definition, a specific drawing entity, the current selection list, the independent model geometry, or the entire model/view/drawing structure.
Extract_Entity.scr	This script extracts a single entity (by DE number, subfigure definition name or network subfigure definition name) or the current selection list, copies the entity to another model, and then optionally writes the model out as an ASCII IGES file.
Free_Display.scr	This script allows you to free the display associated with a specified model.
Free_Model.scr	This scripts allows you to free a previously allocated model.
Help.scr	This script provides on-line help to the various IGES/Works commands.
Highlight_Ent.scr	This script enables you to graphically highlight an entity based on its DE number or entity type.
List_All.scr	This script lists all entities in the current model.

List_Attr.scr	This script cycles the network subfigure instance (type 420) entity in the specified model and gives a listing of any attributes from Table 9 group 4 of the attribute table definition (type 322) entity. Such as would be created by the script Attr_Demo.
Merge_Model.scr	This script copies the contents of a model to another model. Before copying it, it will check the name of subfigures in the source model to see if a subfigure by that name already exists in the target model. If there is one, the subfigure in the source model will be renamed.
Project_IGES.scr	This script flattens a model from 3D to 2D.
Query_Colors.scr	This script cycles through all the entities and makes a record of the colors{XE "Color:using a script to make a record of the colors"}. The script then lists this record.
Query_Curves.scr	This script enables the user to get information on Curves and Surfaces within the Current Model.
Query_Dialog.scr	This script is a simple example of the <i>get dialog</i> command. It access other scripts based on user input.
Query_Drawings.scr	This script enables the user to get information on Drawings and Views within the Current Model.
Query_Fonts.scr	This script cycles through all the entities and makes a record of the fonts. The script then lists this record.
Query_Levels.scr	This script lists the level utilization within the current model.
Query_Subfigures.scr	This script enables the user to get information on Subfigures within the Current Model.
Query_TMs.scr	This script lists the transformation matrices used in the current model.

Read_IGES.scr	This script allows you to read in an ASCII IGES file.
Reduce_Degree.scr	This script changes the degree of NURBS curves and surfaces. It assumes that a model is current and loaded.
Select_Entity.scr	This script enables you to select (pick) an entity and list its IGES information.
Select_Fixed.scr	This script selects and highlights all entities that were fixed by the validation{XE "Validation:using in a script"} procedure. It assumes that the current model has been validated and displayed. To clear the highlighting, use the <i>clear highlight</i> command.
Select_Invalid.scr	This script selects and highlights all entities that were flagged as invalid by the validation{XE "Validation:using in a script"} procedure. It assumes that the current model has been validated and displayed. To clear the highlighting, use the <i>clear highlight</i> command.
Select_Parents.scr	This script selects, highlights and lists the parents of entities in the current selection list. To clear the highlighting, use the <i>clear highlight</i> command.
Unblank_Ents.scr	This script lists all the blanked entities in the file and then optionally unblanks them.
Validate_IGES.scr	This script validates{XE "Validation:using in a script"} the current model. It assumes that the current model is loaded with an IGES file. It creates a log file named model.val (where model is the name of the IGES/Works model) which contains all errors and warnings.

Validate_Model.scr	This script validates{XE "Validation:using in a script"} a model. It presents you with menus of options concerning what model to validate, whether or not to produce a log file{XE "Log file:in a validation script"} and which specific validation options you desire.
Write_IGES.scr	This script formats the contents of the current model to a user specified ASCII IGES file.
convert_trim.scr	Convert {XE "Conversion:of NURBS based trimmed surfaces "}NURBS based trimmed surfaces to parametric spline (type112/114) based trimmed surfaces.
create_userprop.scr	This script helps you create user-defined properties.
decomp_trim.scr	Decompose Trimmed Surfaces{XE "Decomposition:of Trimmed Surfaces, script"}. This script deletes the trimmed surface (type 144/142) entities{XE "Deleting entities:trimmed surface entities, script"}, and leaves the underlying base surface and trimming curves.
modify_matrix.scr	This script changes all entities in the current selection list to point to the input matrix. If an entity already points to a matrix, then the matrix will be modified to point to the input matrix. This routine recursively calls itself in cases of nested matrices.
project_front.scr	This script creates an identity matrix and a view which points to it and projects the current model through this new view. The results are placed in a model named new_model.

read\_files.scr

The script reads in a series of IGES files. The file names to be read in are listed in a file in the form:

```
myfile.igs
```

```
file2.igs
```

```
#this is a comment
```

```
file3.igs
```

## Appendix C. Tolerances

Several tolerance {XE "Tolerances:values"} values are used in IGES/Works for purposes of:

- display
- validation{XE "Validation:tolerance values"}
- conversion{XE "Conversion:tolerance values"}
- subset generation (MAP option)

Some tolerances you will not be as aware of, unless a message, say from validation, is received, or your display does not look as you expected.

### **Note:**

*Other explicit tolerances, on commands, are also used. These are explained in the IGES/Works Command Reference Manual with the command.*

*Often the tolerance given with a command is used as the maximum distance that an approximated line can be off the original curve and still be considered a valid approximation.*

### ZERO\_TOL

A value is considered equal to zero when its absolute value is less than this tolerance.

Default value: 1.0e-13

You may want to change this value based on the requirements of the application that will receive the data.

### EPSILON\_TOL

Two floating point numbers are considered equivalent when the difference between them is less than or equal to this tolerance.

Default value: 1.0e-8

You may want to change this value if the global minimum resolution, expressed in the Global Section of the IGES file, is smaller than this value.

### COLLINEAR\_TOL

Three points are considered collinear based on this tolerance.

Default value: 1.0e-7

You may want to change this value based on the requirements of the application that will receive the data.

### NORMAL\_MAG\_TOL

A normal is considered to have a zero magnitude when the magnitude is less than this tolerance.

Default value: 1.0e-6

You may want to change this value based on the requirements of the application that will receive the data.

## COPLANAR\_TOL

Four or more points are considered coplanar based on this tolerance.

Default value: 1.0e-8

You may want to change this value based on the requirements of the application that will receive the data.

## ANGLE\_TOL

Two angles are considered to be the same when the angular difference between them is less than this tolerance.

Default value: 1.0e-5

You may want to change this value based on the requirements of the application that will receive the data.

## PARAM\_SPACE\_PNT\_TOL

Two parameter space points are considered equivalent when the distance between them is less than this tolerance.

Default value: 1.0e-8

You may want to loosen (make larger) this value if the receiving application does not need a tight tolerance for parameter space curves and the IGES file appears to be failing at the default tolerance value. If the receiving application needs very accurate parameter space curves, the value may need to be tighter (make smaller).

## MODEL\_SPACE\_PNT\_TOL

Two model space points are considered equivalent when the distance between them is less than this tolerance.

Default value: 0.001

You may want to change this value if the receiving application requires a tolerance tighter than the Global Minimum Resolution, expressed in the Global Section of the IGES file. During curve and surface validation{XE "Validation:checks mentioned by error messages"} and conversion{XE "Conversion:checks and error messages"}, certain checks are performed. Some error messages may result which mention the following checks:

- continuity check

The span between each segment of a curve is checked for being continuous. This check is based on the tolerance

## MODEL\_SPACE\_PNT\_TOL.

If the endpoint of one segment does not match the start point of the next segment, within the tolerance, the curve is bad.

You may want to change the tolerance based on the receiving application requirements. Some systems require segments to be continuous with a tolerance of 1.0e-6, others may only need a tolerance of 1.0e-3.

## Glossary

This Glossary explains terms specific to IGES/Works used in this manual and other IGES/Works documentation.

- Alias** An alias is a synonym{XE "Aliases:Glossary definition"} for a command or portion of a command. Frequently typed word combinations can be defined to a short name to speed the use of IGES/Works by reducing the amount of typing required to perform the identical function. IGES/Works defines the alias 'lm' for the command *list model*{XE "Aliases:list model command "} so that entering 'lm' will result in the execution of the *list model* command.
- Current model** The current model is the active model in IGES/Works. If you enter a command, its action will be performed on the current model. Since multiple models may exist in IGES/Works, one and only one may be the current model.
- In the IGES/Works graphical user interface (GUI) the current model is the one allocated for the file in the active File window. If you have multiple IGES files open (more than one model exists in memory), you can make another model current by selecting its File window. See "File" in this Glossary. Refer to Chapter 2, "Learning the basics," in this manual for details on using the GUI. See also, the following commands in the *IGES/Works Command Reference Manual: allocate model, read iges, select model, and display model*.
- Current Selection List** The Current Selection List is the list upon which most IGES/Works selection list commands are performed. The Current Selection List may be saved and accessed later.
- File** In the IGES/Works graphical user interface (GUI) opening a file is the command language equivalent of allocating a model and reading an IGES file into that model. In the GUI, displaying the IGES file is the command language equivalent of displaying the model. See "current model" in this Glossary. Refer to Chapter 2, "Learning the basics," in this manual for details on using the GUI. See also, the following commands in the *IGES/Works Command Reference Manual: allocate model, read iges, select model, and display model*.

Hierarchy	Hierarchy describes an entity relationship conforming to a tree structure. The entity at the top of the hierarchy, the parent entity, is independent. All entities at lower levels in the tree structure, the child entity(s), are dependent entities. Entities may be nested many levels deep. In most cases, the attributes associated with the parent entity take precedence and override the child entity(s) attributes.
IGES File	An IGES file is a disk file in the ASCII format. It is usually created by an IGES translator within a CAD system. An IGES file can be read into an IGES/Works model and a model can be formatted to an IGES file.
Model	A model is simply a collection of IGES entities within IGES/Works. A model may be an IGES file that you read into the system, a set of entities that you copied from an IGES file, or a set of entities that you are creating. You can have multiple models within the software at the same time. Models are identified by a name so you can keep track of what is in each model. See "File" in this Glossary.
Saved Selection List	A saved selection list is a selection list that was originally a Current Selection List but was saved for future reference. A saved selection list can be restored as the Current Selection List.
Script	A script is a file containing a collection of IGES/Works commands which perform some function. Users typically create scripts tailored for their particular function or company. Scripts can be used to flavor IGES files between specific CAD systems, extract data from an IGES file, simplify your analysis and modifications of IGES files, or a variety of other operations.
Selection List	A selection list is a collection of one or more entities in a model. For example, a selection list may be all entities in a model, all line entities in a model, all circular arc entities that reside on layer seven with a radius less than 5, or all surfaces with an area less than a specified size. These lists are a powerful way to access a specific entity or set of entities which possess some common characteristic(s). You may perform many operations on the Current Selection List such as listing entities or copying entities in the list to a new file{XE "Copying entities:using the Current Selection List"}. See "Current Selection List" in this Glossary. Refer also to Chapter 4, "Working with entities," for additional information.

