ERP- Oracle Apps - II

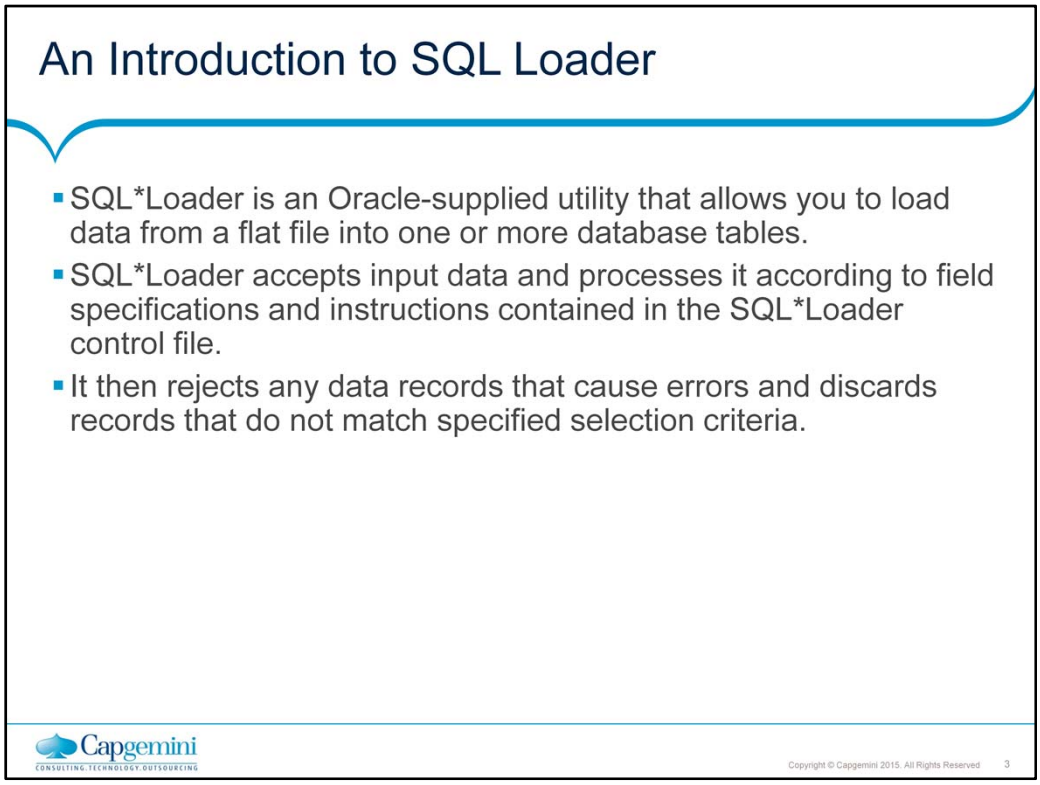
Lesson 14: SQL Loader

Lesson Objectives

- To understand the following:
  - Sql Loader
  - Control File
  - Control file syntax
  - Log File
  - Discard and Bad file

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2



SQL*Loader allows the insertion, appendage, or replacement of records from a sequential text file
SQL*Loader is executed from the operating system prompt, not the SQL prompt.
There must be at least one input file specified for SQL*Loader: The control file
Optionally the raw data may be placed in a separate file.
Output files include:

- Error Log File
- Bad File (Rejected records - Optional)
- Discard File (Does not meet when clause Optional)

Accessing SQL*Loader

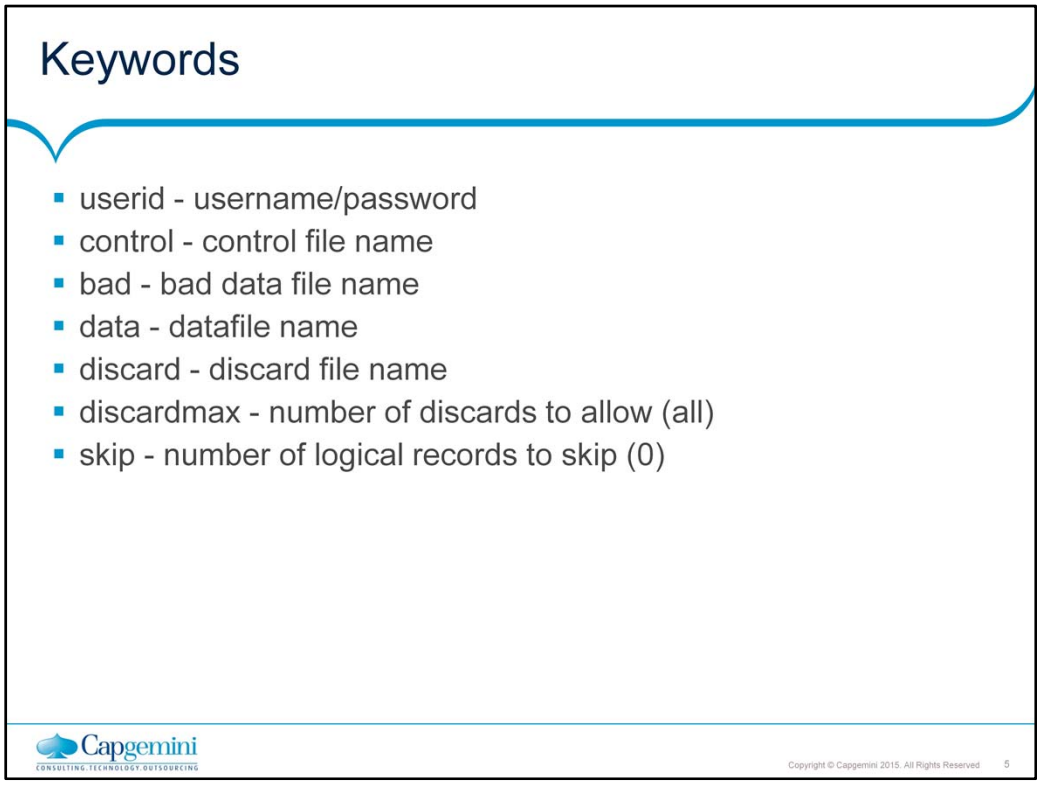
$ sqlldr keyword=value [, keyword=value]
Warning: SID is a reserved word in the SQL*Loader

An Introduction to SQL Loader (contd..)

- SQL*Loader generates a log file that documents its actions. It then passes the prepared input data to the Oracle server.
- The Oracle server rejects any data that causes Oracle errors and inserts the rest into the specified table or tables within the database.
- The basis for almost everything you do with SQL*Loader is a file known as the control file.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4



If the values are specified in the above order, the keywords are not necessary

If the values are specified out of sequence, the keywords must be used.

Delimiters are either commas or spaces

Any values not specified in the control line will either use the default value or be prompted for

Examples
Orca: sqlldr scott/tiger c.ctl l.log b.bad d.dat d.dis
5 0 2000 999

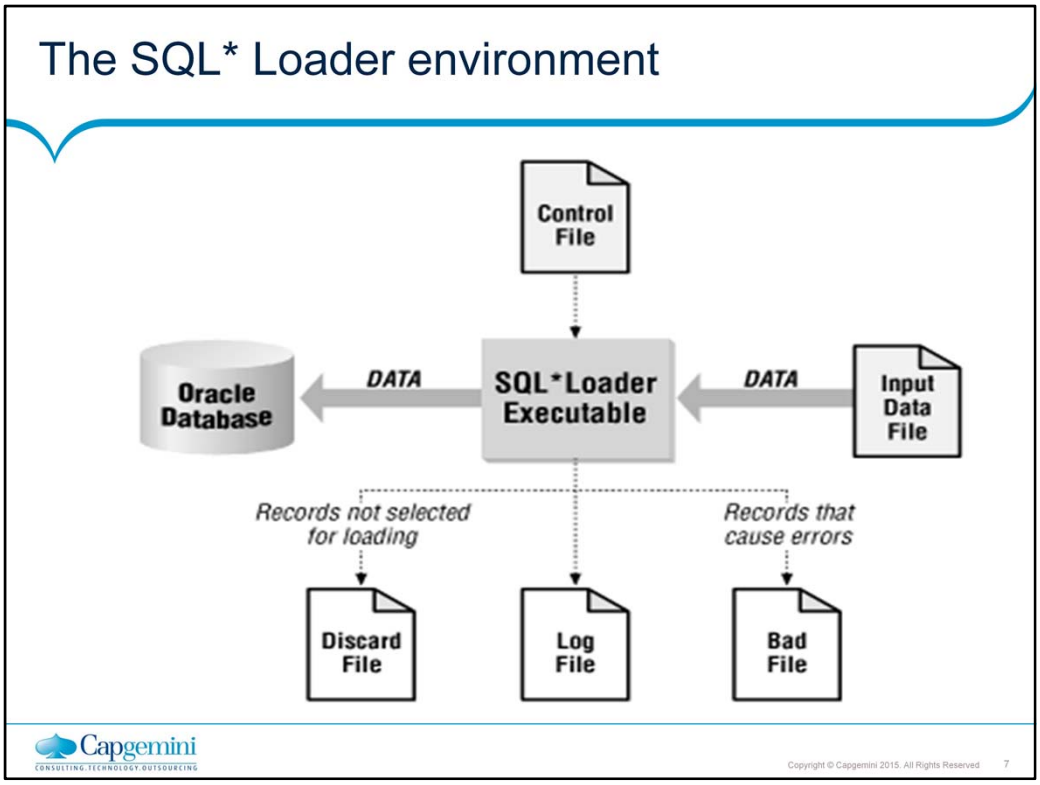
Orca: sqlldr skip=1000,userid=scott/tiger,log=l.log,
control=c.ctl,bad=b.dat

Keywords (contd..)

- load - number of logical records to load (all)
- errors- number of errors to allow (50)
- rows - number of rows to bind array (64)
- bindsize - size of bind array (nnnnn)
- silent - suppress messages during run

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

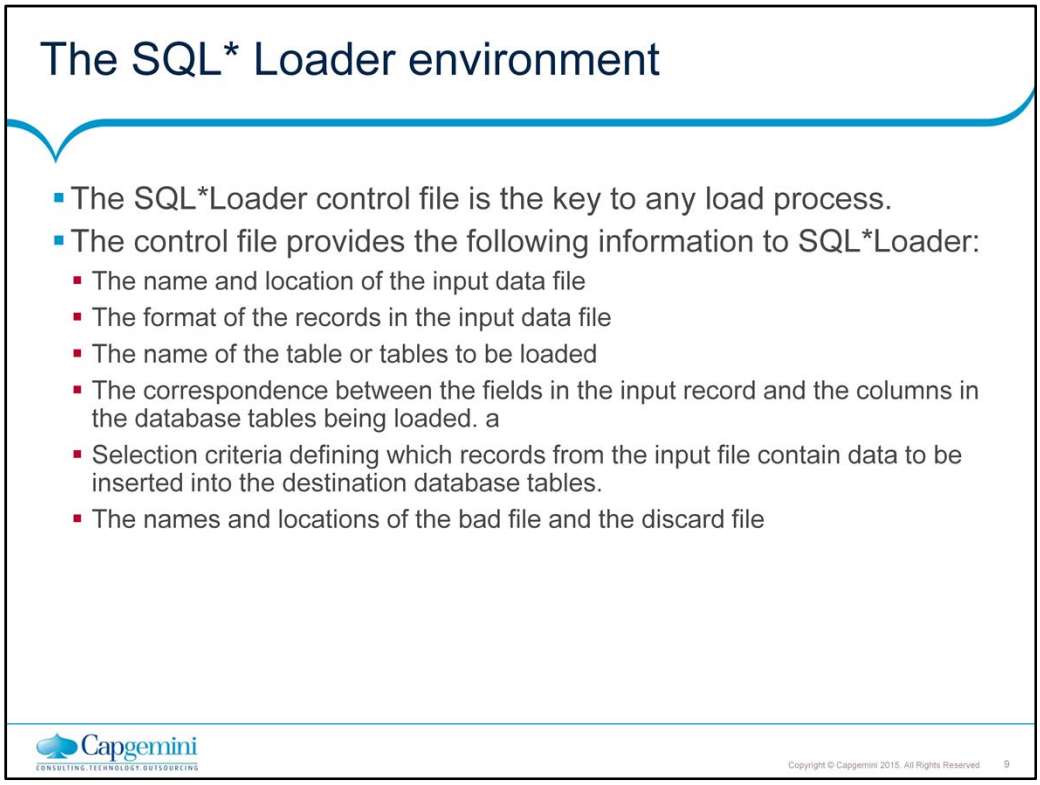

The SQL* Loader environment
Control File
Oracle Database
DATA
SQL*Loader Executable
DATA
Input Data File
Records not selected for loading
Records that cause errors
Discard File
Log File
Bad File
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
7

The SQL* Loader environment

- The SQL*Loader executable does the work of reading the input file and loading the data.
- The input file contains the data to be loaded, and the database receives the data.
- Output of the SQL*Loader is an Oracle database (where the data is loaded), a log file, a bad file, and potentially a discard file.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8



Control files are created via the system text editor.
Control file commands are “free field”
Control file commands may be in upper or lowercase
Comments can be inserted by preceding them with a double dash (--)
SQL*Loader keywords can be enclosed in quotes if you want to use them as table or column names.

```
OPTIONS (options)
 LOAD [DATA]
   [{INFILE | INDDN} {filename | *}
   [STREAM | RECORD | FIXED len |
   BLOCKSIZE size] | VARIABLE [len] ]
   [{BADFILE | BADDN }  filename]
   [{DISCARDFILE | DISCARDDN } filename]
   [{DISCARDS | DISCARDMAX} n]

   [{INDDN | INFILE} . . .] [APPEND | REPLACE | INSERT]
   [RECLEN n]
   [{CONCATENATE n |
     CONTINUEIF {[THIS | NEXT] (start [:end])
    | LAST}
       operator {‘char-str’ | x’hex-str’} } ]
```

```
INTO TABLE tablename
    [APPEND | REPLACE | INSERT]
    [WHEN field-condition [AND
    field-condition  . . .]]
    [FIELDS [delimiter-spec]]
   (column-name
    {RECNUM | CONSTANT value |
    SEQUENCE ({n | MAX | COUNT}
    [, increment] ) |
    [POSITION ({start[:end] | * [+n] } ) ]
    datatype-spec
    [NULLIF field condition]
    [DEFAULTIF field-condition]}
    [, . . .] )
   [INTO TABLE]     [BEGINDATA data ]
```

The SQL* Loader control file (contd..)

- It's also possible for the control file to contain the actual data to be loaded. This is sometimes done when small amounts of data need to be distributed to many sites, because it reduces (to just one file) the number of files that need to be passed around. If the data to be loaded is contained in the control file, then there is no need for a separate data file.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

```
Ex:-  LOAD DATA  INFILE * INTO TABLE dept   FIELDS TERMINATED BY ','
     OPTIONALLY ENCLOSED BY ""“
     (deptno, dname, loc)
     BEGINDATA
      12,RESEARCH,"SARATOGA"
      10,ACCOUNTING,”CLEVELAND”
      11,"ART",”SALEM”
       13,FINANCE,"BOSTON“


Options: LOAD, SKIP, ERRORS, BINSIZE, SILENT
 Example OPTIONS (SKIP=100, LOAD=200)

 Field Condition:
  {(start [:end]) | column name } operator
   {‘char string’ | X’hex byte string’}
 Example: NAME = ‘Bill Collector’
```

Datatype Spec:
CHAR [(length)] [delimiter spec]
DATE [(length)] [mask] [delimiter spec]
VARCHAR [(length)]
SMALLINT
DOUBLE
GRAPHIC [EXTERNAL] [(length)]
{INTEGER | FLOAT | DECIMAL} EXTERNAL [(length)]
INTEGER [(length)]
FLOAT [(length)]
DECIMAL (digits [,precision])
VARGRAPHIC [(length)]
Delimiter Spec:
[TERMINATED [BY] {WHITESPACE | [X] 'char'}]
[[OPTIONALLY] ENCLOSED [BY] [X] 'char']

Control File Syntax Notes

- Specifying Data Files
- Must use the filename statement optionally followed by the File Read Mode
- Filename: To specify the name of the file containing the raw data.
- {INFILE | INDDN} {filename | *} filename: The name of the file containing the data.
- *: The data is in the control file

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

Control File Syntax Notes (contd..)

- Read File Mode:
- Tells SQL*Loader how to open the data file and retrieve physical records from it. This provide a great deal of flexibility on a wide variety of Operating Systems

- STREAM | RECORD |
- FIXED len [BLOCKSIZE size] VARIABLE [len]

- Specifying Multiple Data Files:
  - INFILE DATA1.DAT
  - INFILE DATA2.DAT

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

Control File Syntax Notes (contd..)

- Specifying Bad Files: (records with formatting errors or ORACLE errors)
- [BADFILE | BADDN] filespec
- Specifying Discard Files: (Records which meet none of the filtering requirements)
- [DISCARDFILE | DISCARDDN] filespec

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

Control File Syntax Notes (contd..)

- Specifying The Position of a Data Field
- POSITION ({start[:end] | * | [+n] })
- Start: Starting column number
- End: Ending column number. If omitted the length of the datafield is determined from the datatype in the datafile.
- Maybe specified: start:end or start-end

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

Specifying Control File Syntax Notes (contd..)

- * : Indicates that the data field follows immediately after the previous field. If an * is used for the first data field, that data field is assumed to begin in datatype format.
- +n : An offset, specified as +n, may be used with * to offset this field from the previous one.
- POSITION may be omitted entirely. If so, the position specification for the data field is the same as if POSITION (*) had been used.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17

Examples:

```
SITEID  POSITION(*) SMALLINT
SITELOC POSITION(*) INTEGER
ENAME POSITION (1:20) CHAR
EMPNO POSITION (22-26) INTEGER EXTERNAL
ALLOW POSITION (*+2) INTEGER EXTERNAL TERMINATED BY “/”
```

The Log file

- The log file is a record of SQL*Loader's activities during a load session. It contains information such as the following:
  - The names of the control file, log file, bad file, discard file, and data file.
  - The values of several command-line parameters
  - A detailed breakdown of the fields and datatypes in the data file that was loaded.
  - Error messages for records that cause errors.
  - Messages indicating when records have been discarded.
  - A summary of the load that includes the number of logical records read from the data file, the number of rows rejected because of errors, the number of rows discarded because of selection criteria, and the elapsed time of the load.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 18

The Bad File and the Discard File

- Whenever you insert data into a database, you run the risk of that insert failing because of some type of error such as violating the Integrity Constraint.
- Whenever SQL*Loader encounters a database error while trying to load a record, it writes that record to a file known as the bad file.
- Discard files, on the other hand, are used to hold records that do not meet selection criteria specified in the SQL*Loader control file.
- Records that do not meet the specified criteria are not loaded, and are instead written to a file known as the discard file.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19

Conventional Path Load VS Direct Path Load

- SQL*Loader provides two methods to load data: Conventional Path, which uses a a SQL INSERT statement with a bind array, and Direct Path, which loads data directly into a database. Direct path load is much faster than conventional path load.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20

Summary

- In this lesson we have covered:
  - Sql Loader
  - Control File
  - Control File syntax
  - Log File
  - Discard and Bad file

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

Summary

- In this lesson, you should have learned how to:
  - What is an Interface?
  - Interface Types
  - Importing Information into Oracle Financials Applications
  - Open interfaces
  - API Interfaces
  - Integration
  - The benefits of using open interfaces
  - The Oracle Applications Open Interface Model
  - How to manage your open interface processing

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 22

Review Question

- Question 1: SQL*Loader is an Oracle-supplied utility that allows you to load data from a _____ into one or more
- database tables.
  - Option 1: Control file
  - Option 2: Flat file
  - Option 3: Log file
  - Option 4: Discard file
- Question 2. The basis for almost everything you do with SQL*Loader is a file known as the _____
  - Option 1: Control file
  - Option 2: Flat file
  - Option 3: Log file
  - Option 4: Discard file

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

ERP- Oracle Apps

Lesson 15: Object Standards

Lesson Objectives

- At the end of the session you will be able to:
  - Understand the correct usage of data types
  - Work with Oracle Applications Views
  - Sequences
  - Understand Table Registration
  - Know What are WHO columns

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

Data Types to be avoided

- Avoid creating tables with the LONG, LONG RAW, or RAW datatypes
- Within Oracle Forms, you cannot search using wildcards on any column of these types. Use VARCHAR2(2000) columns instead

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

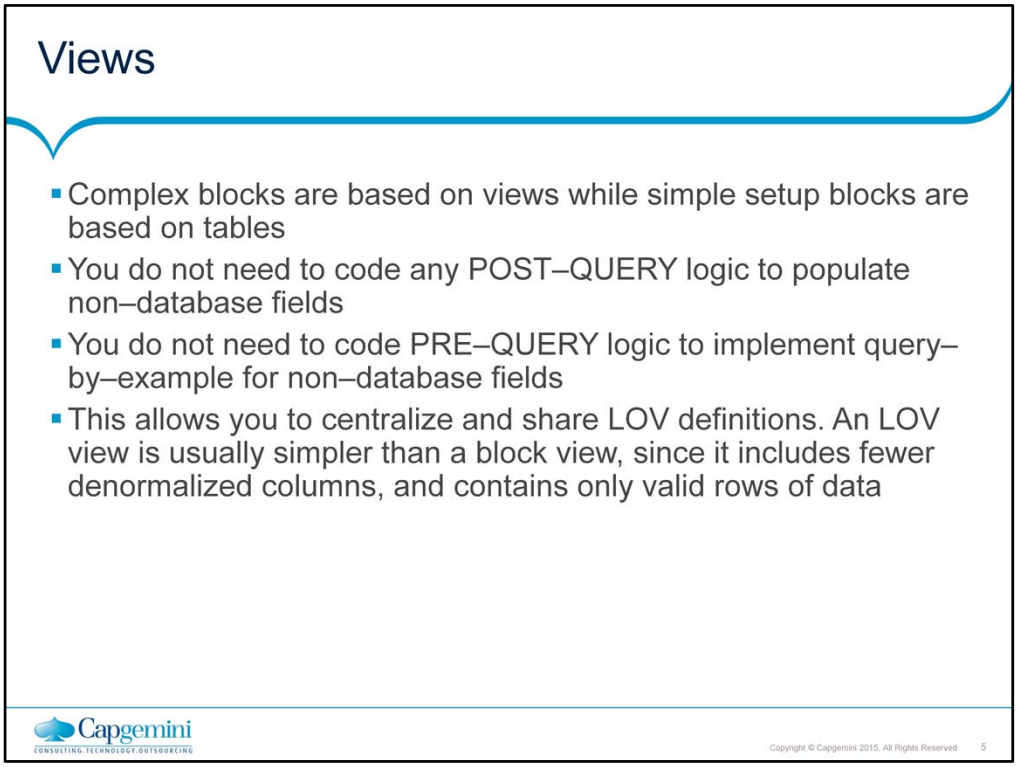
 3

Examples of wrong data type usage

- Setting a value ‘Generator’ to a variable/field whose field size is less than 9 would raise an exception
- Wrong format specification in the type conversion functions like to_date or to_char would raise an exception
- Storing a value 123.4567 in a field having two decimal precision and using it in the further calculations would result in unexpected result

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4



When you use a Post-Query trigger to SELECT non-base table values into control items, Form Builder marks each record as CHANGED, and so fires the When-Validate-Item trigger by default. You can avoid the execution of the When-Validate-Item trigger by explicitly setting the Status property of each record to QUERY in the Post-Query trigger. To set record status programmatically, use SET_RECORD_PROPERTY.

Views (Contd…)

- You should also base your Lists of Values (LOVs) on views
- Whenever performance is an issue and your table has foreign keys, you should define a view to improve performance
- Views allow a single SQL statement to process the foreign keys, reducing parses by the server, and reducing network traffic

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

Views: Advantages

- Views are extremely desirable because:
  - They speed development, as developers can build on logic they already encapsulated
  - They modularize code, often meaning that a correction or enhancement can be made in a single location
  - They reduce network traffic
  - They are often useful for reporting or other activities
  - They can be easily and centrally patched at a customer site

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

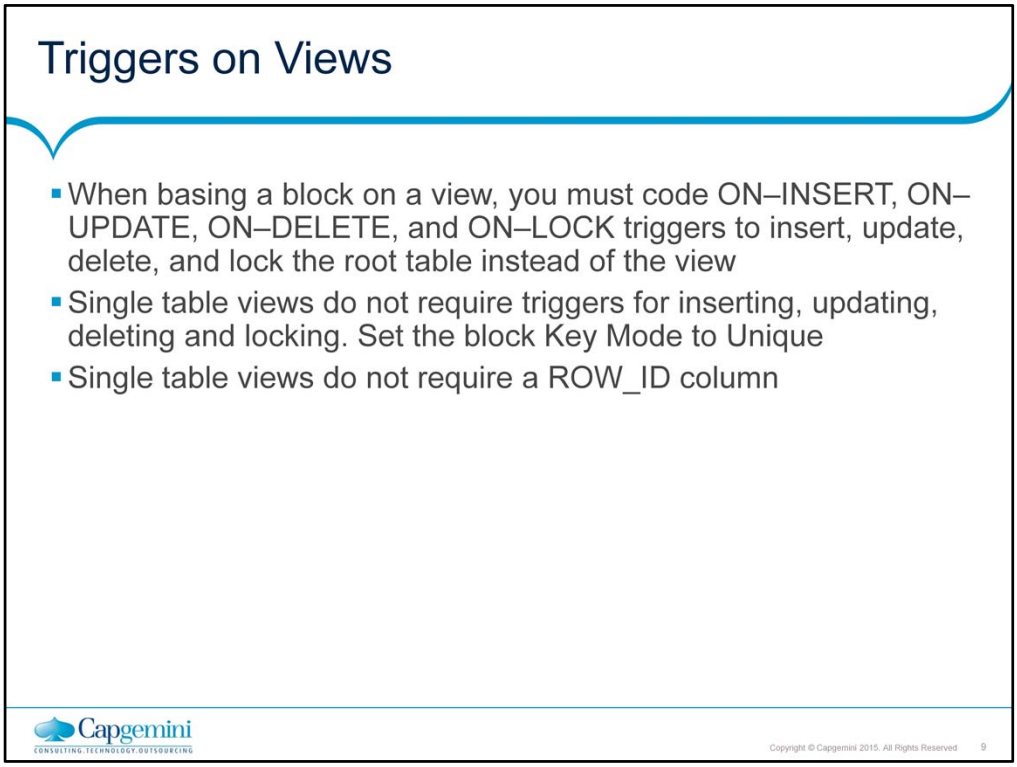
 7

Views: Restrictions

- Avoid creating views that are used by only one SQL statement
- Creating a view that is only used by a single procedure increases maintenance load because both the code containing the SQL statement and the view must be maintained

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8



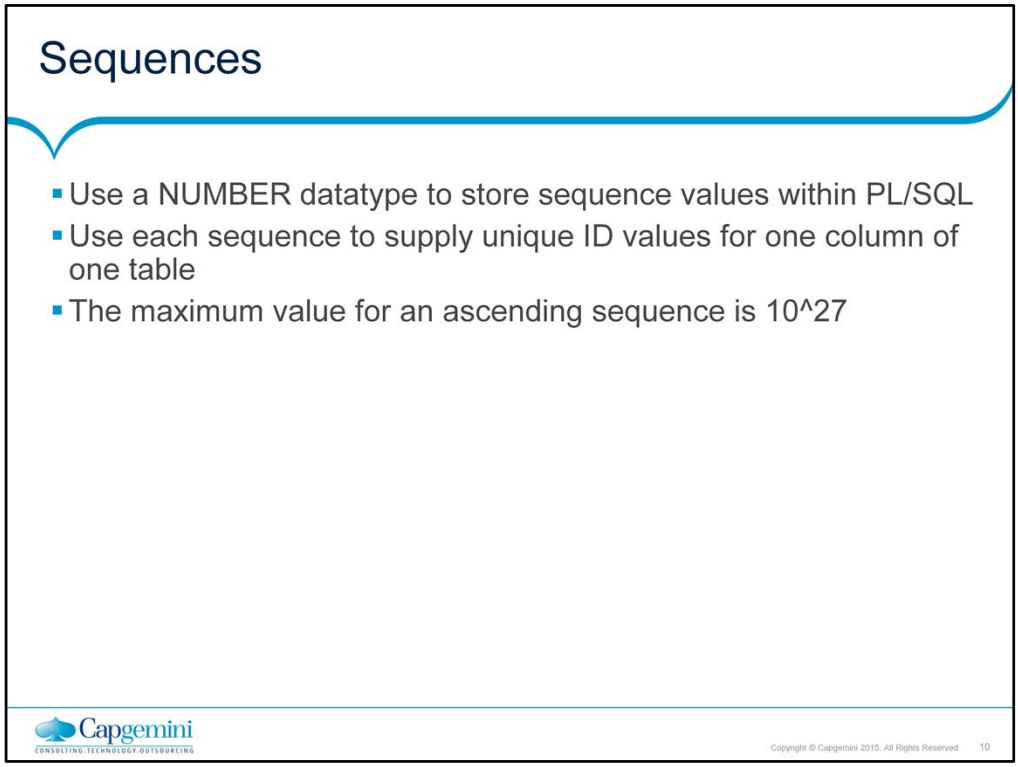
Your view should then include all of the columns in the root table, including the WHO columns, and denormalized foreign key information.

Suggestion: You only need to include the ROWID column if an Oracle Forms block is based on this view. The Oracle Forms field corresponding to the ROW_ID pseudo–column should use the ROW_ID property class.

Change Block Key Mode
In Oracle Forms, you need to change the block Key Mode property to Non–Updatable to turn off Oracle Forms default ROWID references for blocks based on views. Specify the primary keys for your view by setting the item level property Primary Key to True.

For example, a view based on the EMP table has the columns ROW_ID, EMPNO, ENAME, DEPTNO, and DNAME. Set the Key Mode property of block EMP_V to Non–Updatable, and set the Primary Key property of EMPNO to True.

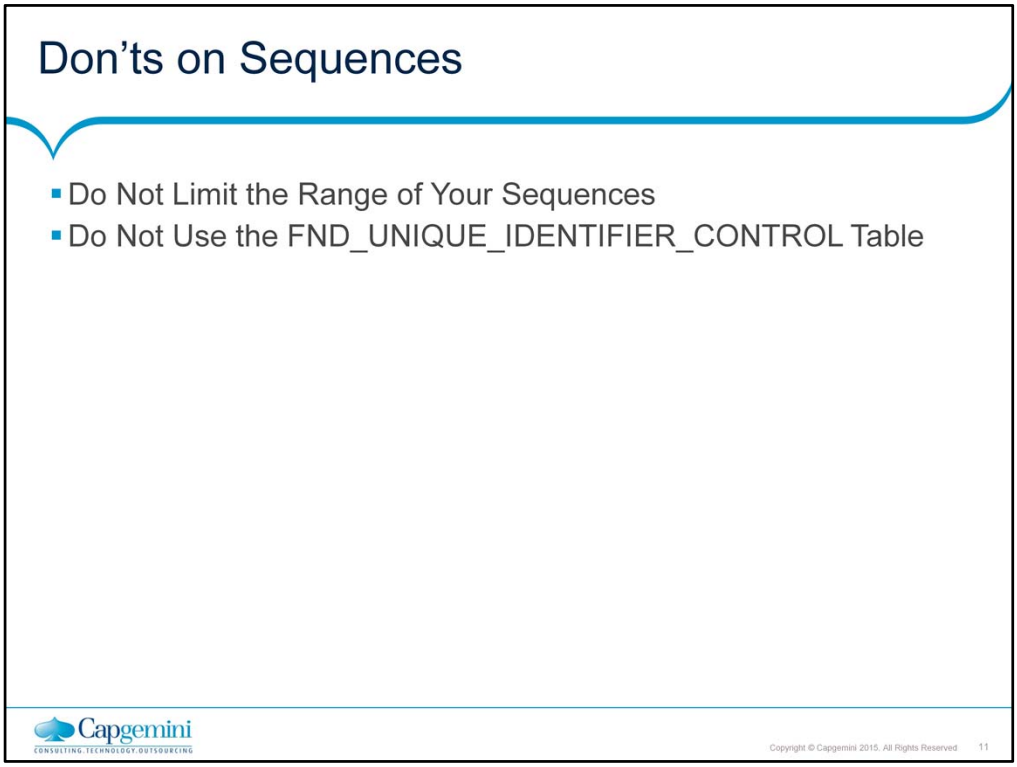
If your block is based on a table, the block Key Mode should be Unique.



Sequences can be created through front end of Apps or at the database level.
Sequences created using the from end can be assigned to a particular field on the form.
Seeded sequences are also available.

The purpose of backend sequences would be to provide unique ids to certain fields, if they are not part of the source data.

For more details, refer to “Sequences” (See Page 3-10) from Oracle Applications Developer’s Guide.



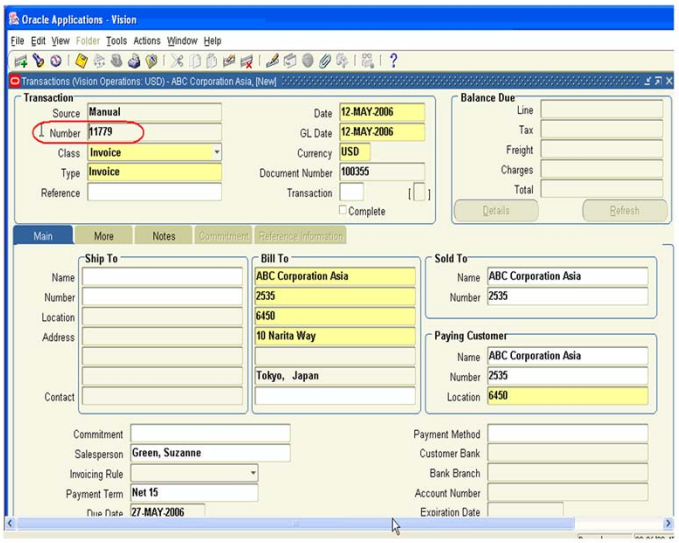
Do not create sequences that wrap using the CYCLE option or that have a specified MAXVALUE. The total range of sequences is so great that the upper limits realistically are never encountered. In general, do not design sequences that wrap or have limited ranges.

Do not rely on the FND_UNIQUE_IDENTIFIER_CONTROL table to supply sequential values. Use a sequence or the sequential numbering package instead. The FND_UNIQUE_IDENTIFIER_CONTROL table is obsolete and should not have any rows for objects in your product. Additionally, do not create application–specific versions of the FND table to replace the FND_UNIQUE_IDENTIFIER_CONTROL table.

Fndutcsq.sql – this script converts rows in FND_UNIQUE_IDENTIFIER_CONTROL to sequences

Create a Sequence

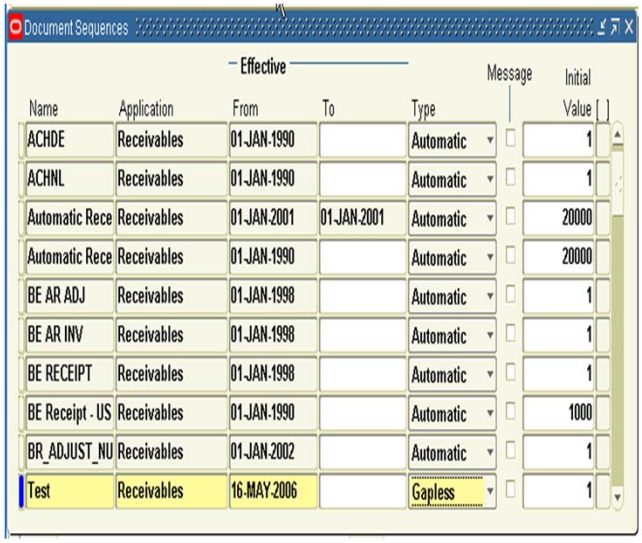
- The Transaction number is generated through a sequence, when the record is saved



To navigate to the form shown above, select Receivables, Vision Operations (USA) -> Transactions -> Transactions. The Transaction number is generated through a sequence.

Create a Sequence (Contd…)

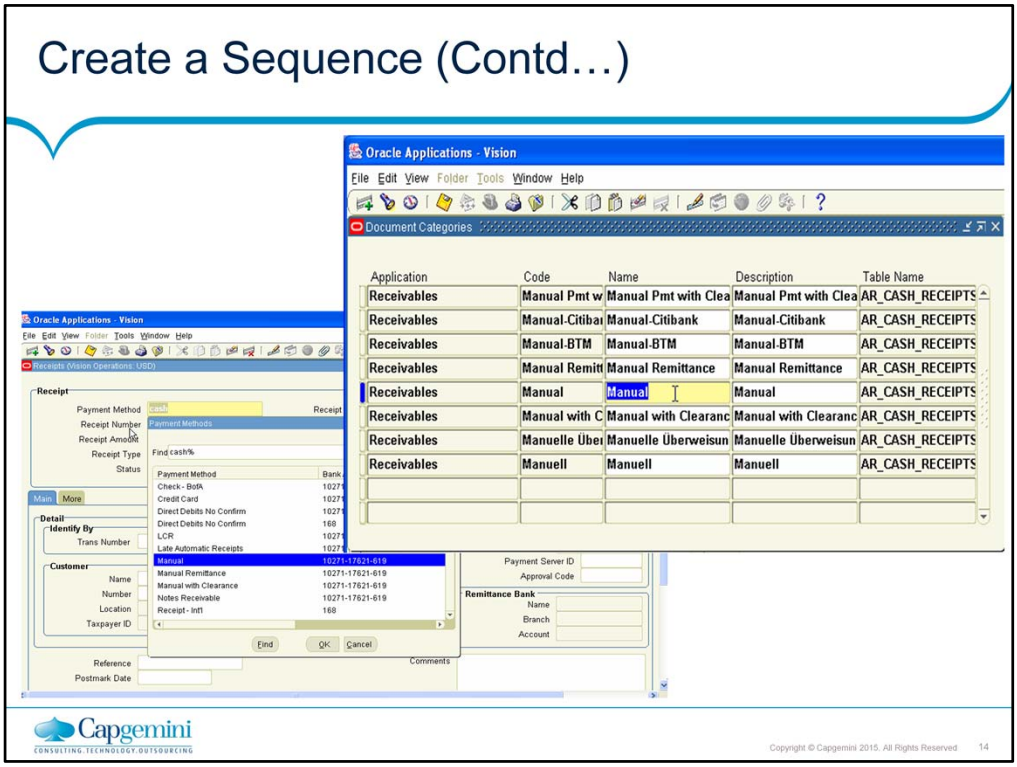
- Shown in the figure are existing sequences.
- You can also create a new sequence
- Navigate to System Administrator -> Document-> Define



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

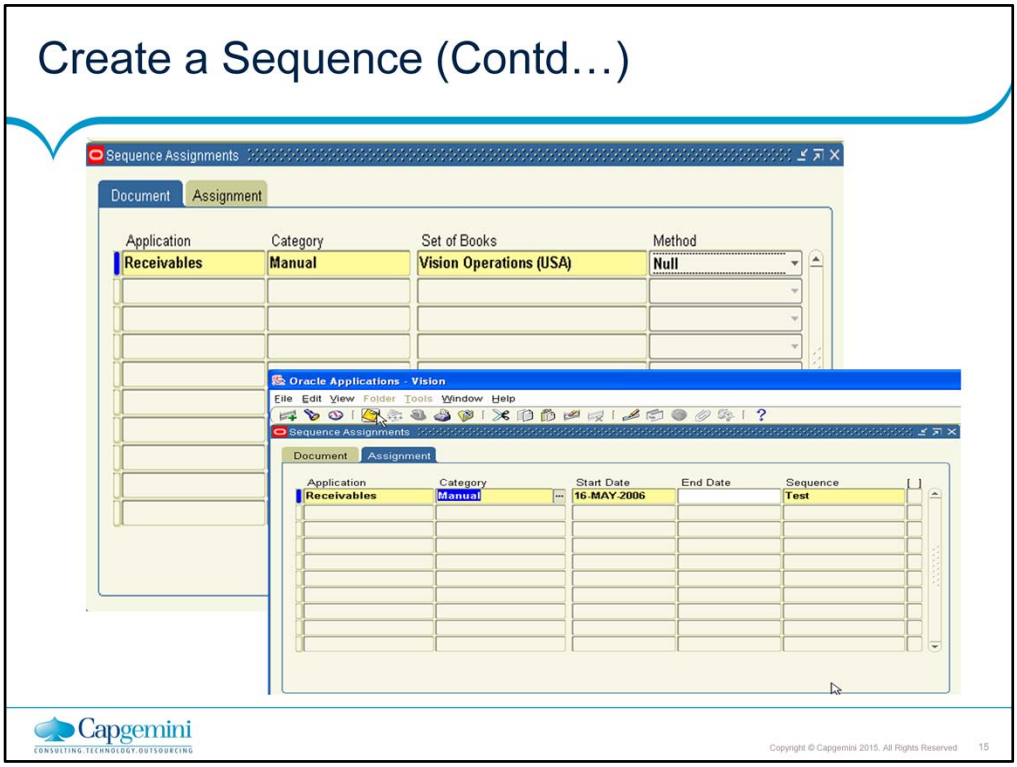
A new sequence Test is created and its type is set to Gapless, i.e. which indicates that there would be no gaps in the sequence numbers.



The window on the top can be accessed by selecting System Administrator -> Document-> Categories. This window displays the categories. Manual payment is one such category, associated with Application : Receivables and Table name : AR_CASH_RECEIPTS_ALL

Various payment Methods (Categories) are as shown in the Receipts form on the left. The Receipts form can be accessed by selecting : Receivables, Vision Operations (USA) -> Receipts.

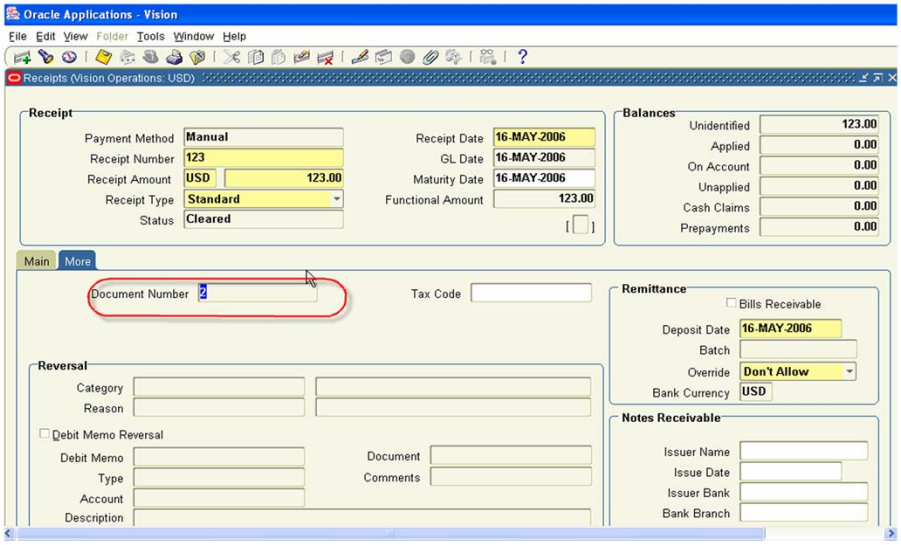
The sequence created is created for manual type payments.



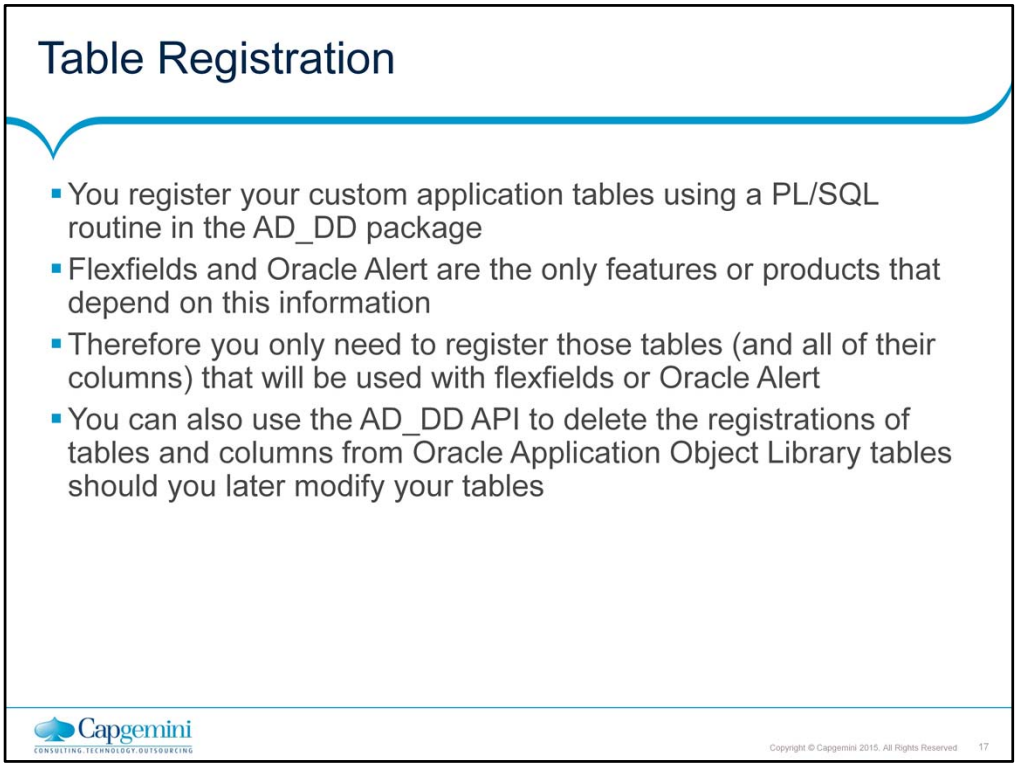
The sequence is assigned to the Manual category in the Sequence Assignments form. This can be accessed by selecting System Administrator -> Document-> Sequence Assignments.

Note that initially the value under Method is set to Null, indicating that no sequence is yet assigned. To assign a sequence, click on the Assignment

Create a Sequence (Contd…)



In the Receipts form, when a record is created for Manual type of payment, the Document Number is automatically generated through the sequence.



Sample Procedures in the Package

```
procedure register_table (p_appl_short_name in
varchar2,
p_tab_name in varchar2,
p_tab_type in varchar2,
p_next_extent in number default 512,
p_pct_free in number default 10,
p_pct_used in number default 70);
```

```
procedure register_column (p_appl_short_name in
varchar2,
p_tab_name in varchar2,
p_col_name in varchar2,
p_col_seq in number,
p_col_type in varchar2,
p_col_width in number,
p_nullable in varchar2,
p_translate in varchar2);
```

Here is an example of using the AD_DD package to register a flexfield table and its columns:

```
EXECUTE ad_dd.register_table('FND',
                  'CUST_FLEX_TEST', 'T', 8, 10, 90);
```

WHO Columns

- The Record History (WHO) feature reports information about who created or updated rows in Oracle Applications tables
- If you add special WHO columns to your tables and WHO logic to your forms and stored procedures, your users can track changes made to their data
- By looking at WHO columns, users can differentiate between changes made by forms and changes made by concurrent programs

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

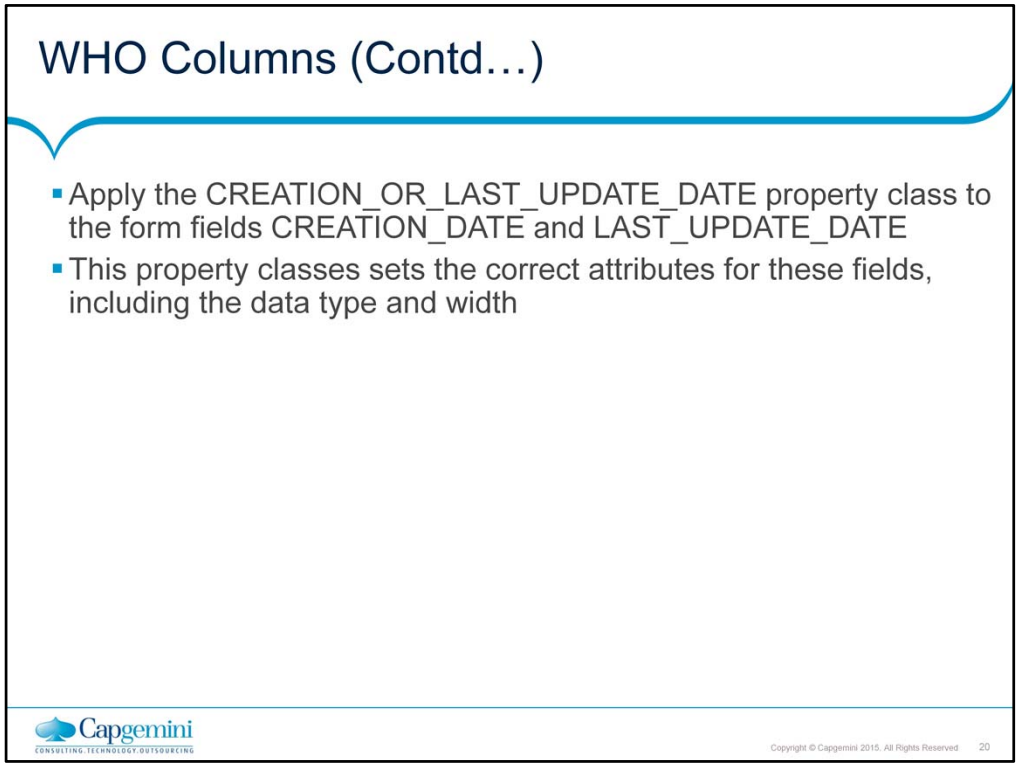
 18

WHO Columns (Contd…)

- You represent each of the WHO columns as hidden fields in each block of your form (corresponding to the WHO columns in each underlying table)
- Call FND_STANDARD.SET_WHO in PRE–UPDATE and PRE–INSERT to populate these fields
- Set the CREATED_BY and CREATION_DATE columns only when you insert a row (using FND_STANDARD.SET_WHO for a form)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

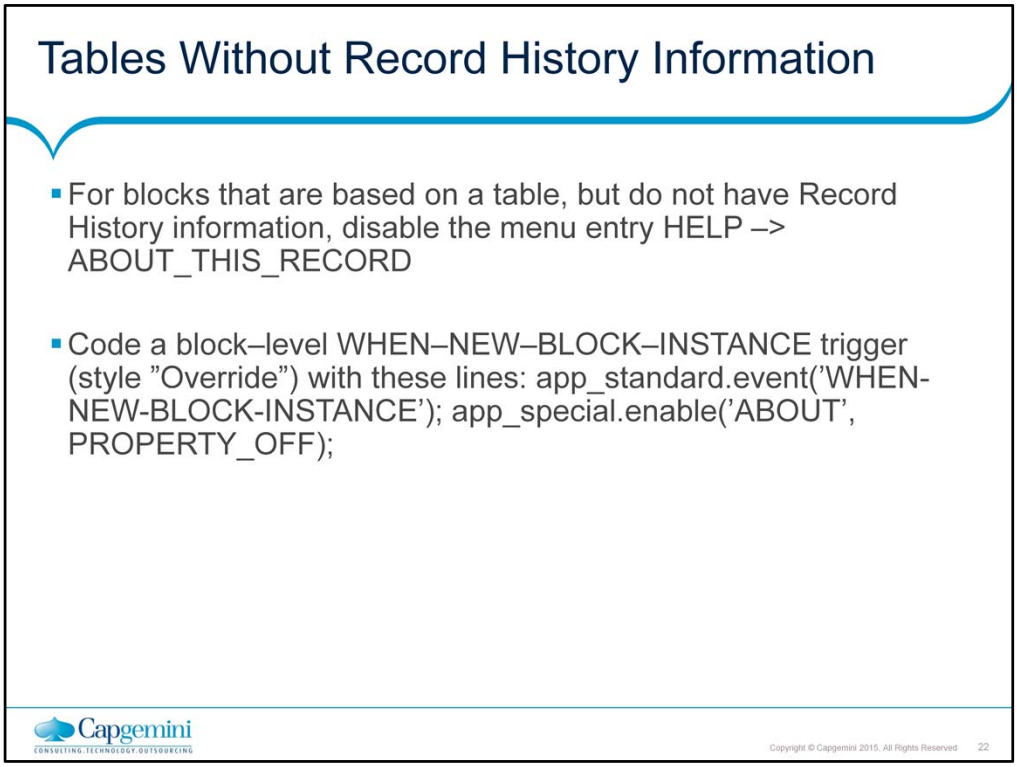
 19



For more details, refer to “Tracking Data Changes with Record History (WHO)” (See Page 3-2) from Oracle Applications Developer’s Guide.

Record History

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING



Use the APP_SPECIAL package to enable and customize menu entries and buttons on the toolbar. You can use the APP_SPECIAL.ENABLE procedure to dynamically control menu items, if the behavior you need is not provided automatically. First, determine if the default menu control handles the menu item in question, and ensure that there really is a need to override the default behaviors.
If the menu item is not controlled by the default menu control, use any appropriate trigger (typically PRE–BLOCK or WHEN–NEW–BLOCK–INSTANCE), adding the code:

```
app_special.enable('the menu item',
PROPERTY_OFF|ON);
```

Turn the menu item back on when you leave (typically POST–BLOCK) by calling:

```
app_special.enable('the menu item',
PROPERTY_ON|OFF);
```

Include the full name of the menu item in this call, for example:

```
app_special.enable('CLEAR.FIELD', PROPERTY_OFF);
```

You can determine the full names of the menu items by copying FNDMENU from the AU_TOP/resource/<language> area and opening the copy to examine the menu items.

If the menu item is controlled by the default menu control and you want to modify its behavior (enable or disable it), create the field– or block–level trigger listed (either WHEN–NEW–BLOCK–INSTANCE, WHEN–NEW–RECORD–INSTANCE, or WHEN–NEW–ITEM–INSTANCE). Set the trigger Execution Hierarchy to ”Override” and add the following code:

```
app_standard.event('TRIGGER_NAME');
app_special.enable('Menu_item',
PROPERTY_OFF|ON);
```

The item will be correctly reset in other blocks by the default menu control, so it is not necessary to reset it when leaving the block, record, or item.

Summary

- In this session we covered:
  - The Correct Usage of Data Types
  - Oracle Applications Views
  - Sequences
  - Table Registration Process
  - What are WHO columns

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 24

Review Question

- Question 3: Whenever SQL*Loader encounters a database error while trying to load a record, it writes that record to a file known as the _____
  - Option 1: Control file
  - Option 2: Flat file
  - Option 3: Bad file
  - Option 4: Discard file

Review Question

- Question 4: Records that do not meet the specified criteria are not loaded and are instead written to a file known as _____
  - Option 1: Control file
  - Option 2: Flat file
  - Option 3: Bad file
  - Option 4: Discard file

ERP – Oracle Apps

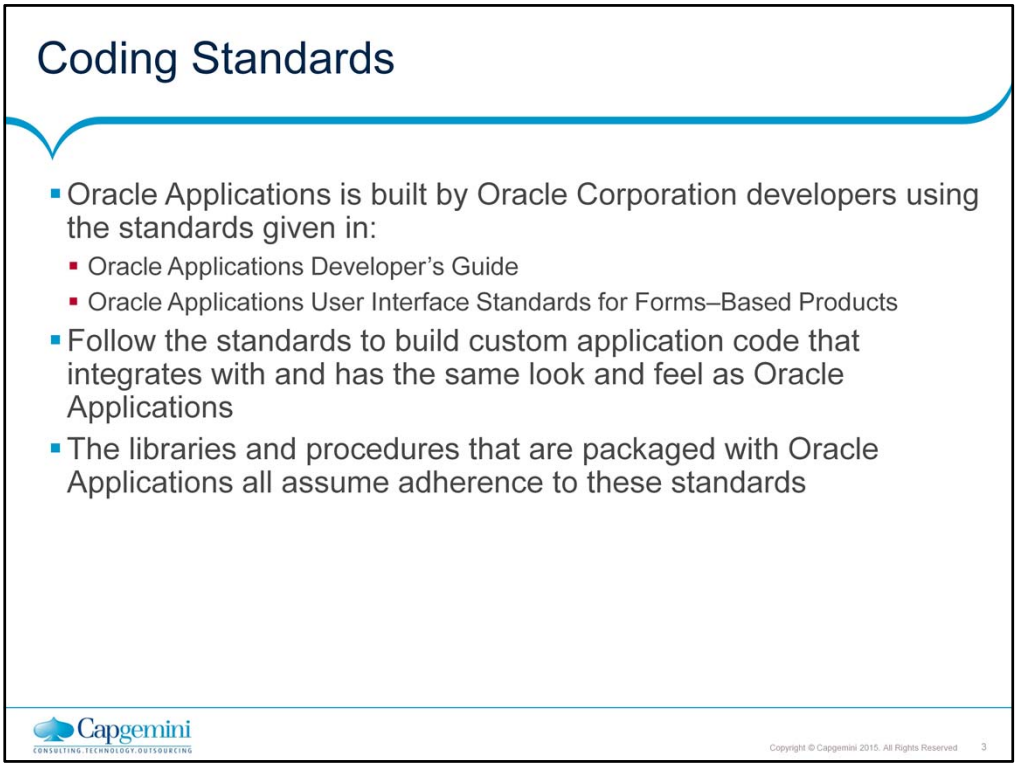
Lesson 16: Coding Standards

Lesson Objectives

- To understand the coding standards of following topics:
  - Basic of Coding Standards
  - Handlers
  - Triggers
  - SQL
  - PL/SQL Code

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

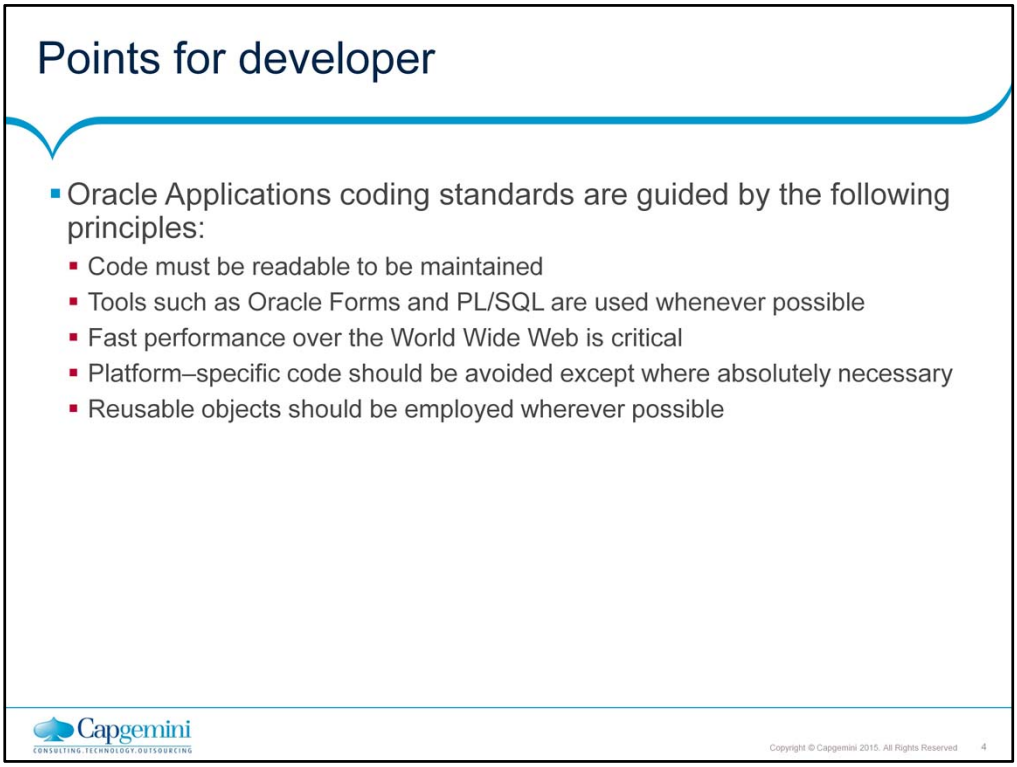
 2



Importance of these Standards

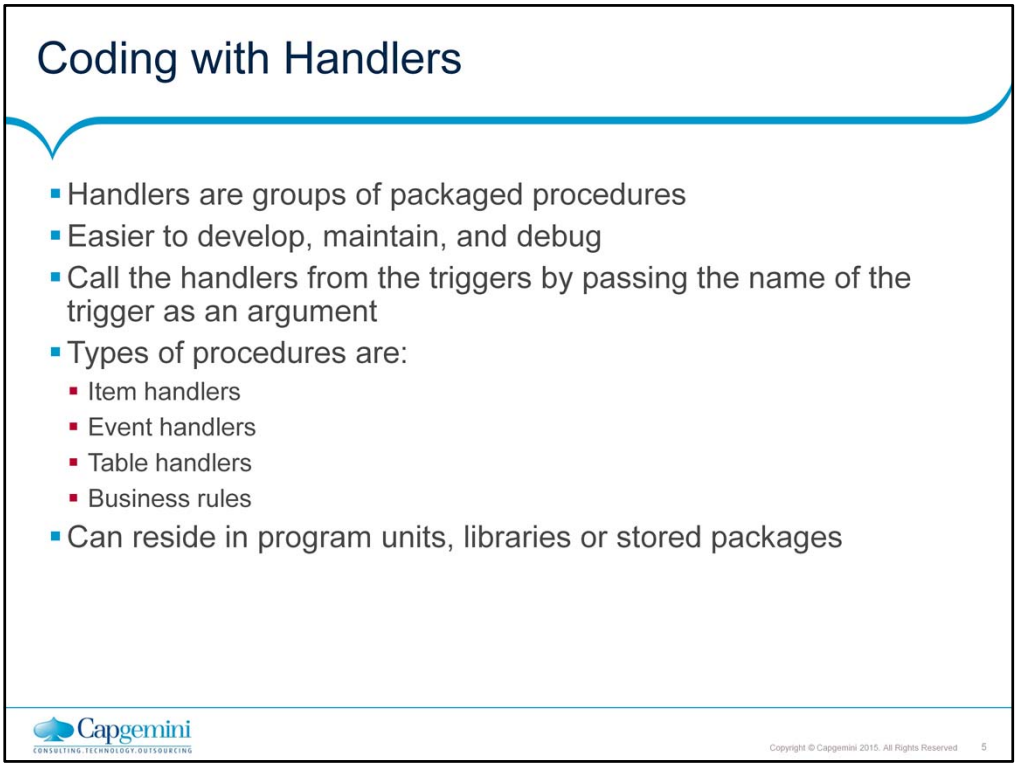
The coding standards described in Oracle Applications Developer's Guide, together with the user interface standards described in the Oracle Applications User Interface Standards for Forms–Based Products, are used by Oracle Corporation developers to build Oracle Applications. If you want to build custom application code that integrates with and has the same look and feel as Oracle Applications, you must follow these standards. If you do not follow these standards exactly as they are presented, you may not achieve an acceptable result.

The libraries and procedures that are packaged with Oracle Applications all assume adherence to these standards. In fact, since the behavior of Oracle Forms, the Oracle Applications standard libraries, and the standards are so tightly linked, a deviation from standards that appears to be minor may in fact have far–reaching and unpredictable results. Therefore, its recommend that when you develop custom application code, you follow the standards exactly as they are described in these manuals.



Oracle Applications coding standards are guided by the following principles:

- Code must be readable to be maintained
- Tools such as Oracle Forms and PL/SQL are used whenever possible (avoid complex user exits using other coding languages)
- Fast performance over the World Wide Web (the web) is critical
- Platform–specific code should be avoided except where absolutely necessary
- Reusable objects should be employed wherever possible



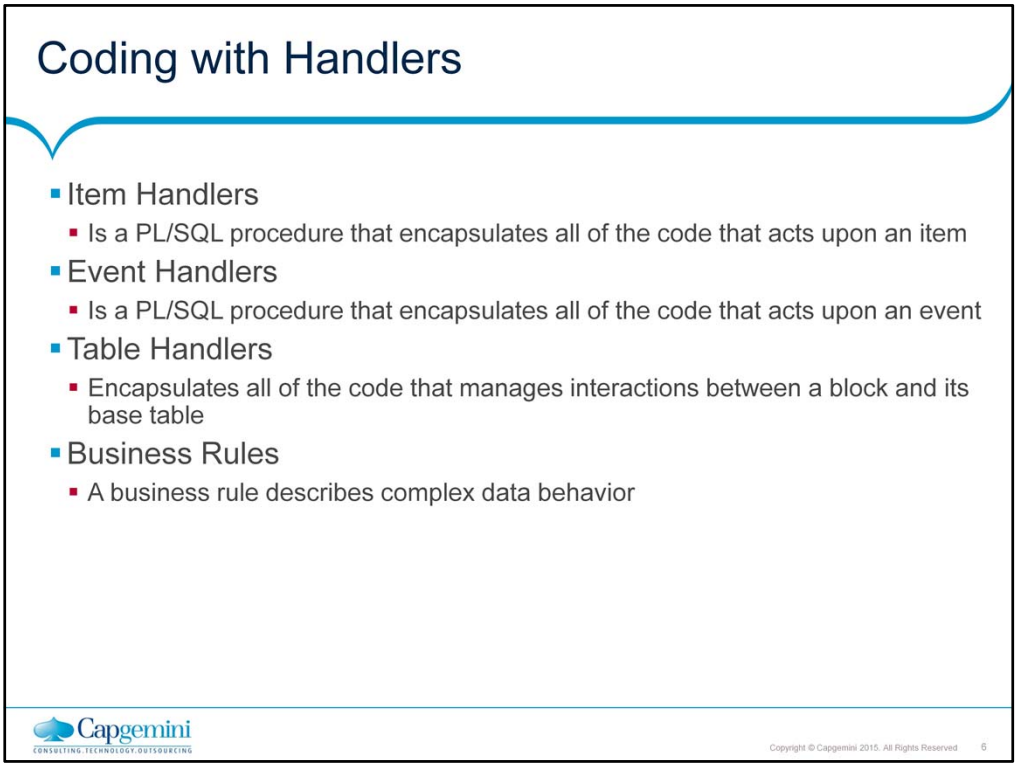
Oracle Applications uses groups of packaged procedures, called handlers, to organize PL/SQL code in forms so that it is easier to develop, maintain, and debug.

In Oracle Forms, code is placed in triggers, which execute the code when that trigger event occurs. Implementing complex logic may require scattering its code across multiple triggers. Because code in triggers is not located in one place, it cannot be written or reviewed comprehensively, making development, maintenance, and debugging more difficult. To determine what code and events affect a particular item, a developer must scan many triggers throughout the form. Code that affects multiple items can be extremely difficult to trace.

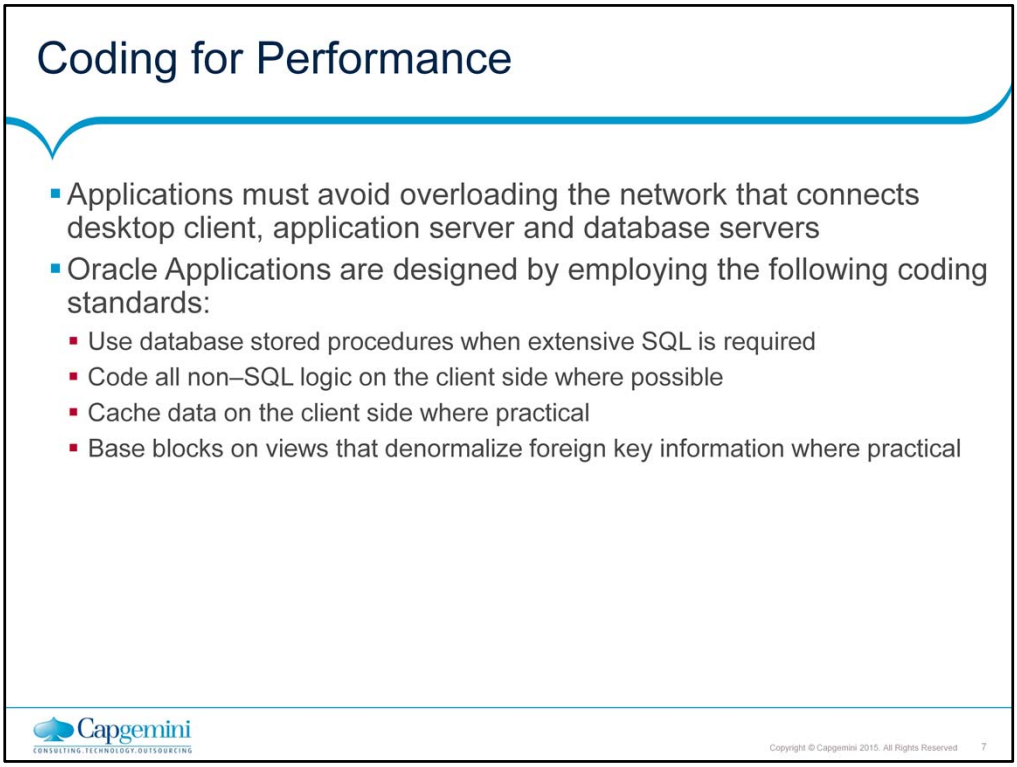
To centralize the code so it is easier to develop, maintain, and debug, place the code in packaged procedures and call those procedures from the triggers. Pass the name of the trigger as an argument for the procedure to process. This scheme allows the code for a single business rule to be associated with multiple trigger points, but to reside in a single location.

There are different kinds of procedures for the different kinds of code you write: item handlers, event handlers, table handlers, and business rules. Code resides in these procedures; do not put any code in the triggers other than calls to the procedures.

Handlers may reside in program units in the form itself, in form libraries, or in stored packages in the database as appropriate.



Item Handlers
An item handler is a PL/SQL procedure that encapsulates all of the code that acts upon an item. Most of the validation, defaulting, and behavior logic for an item is typically in an item handler.
Event Handlers
An event handler is a PL/SQL procedure that encapsulates all of the code that acts upon an event. Usually event handlers exist to satisfy requirements of either Oracle Forms or the Oracle Applications User Interface Standards for Forms–Based Products, as opposed to particular business requirements for a product.
Table Handlers
A table handler encapsulates all of the code that manages interactions between a block and its base table. When an updatable block is based on a view, supply procedures to manage the insert, update, lock and delete. Referential integrity checks often require additional procedures. Table handlers typically reside in the database but may also reside on the forms server depending on size and the amount of interaction with the database.
Business Rules
A business rule describes complex data behavior. For example, one business rule is: ”A discount cannot be greater than 10% if the current credit rating of the buyer is less than ’Good’.” Another business rule is: ”A Need–By Date is required if a requisition is made for an inventory item.”

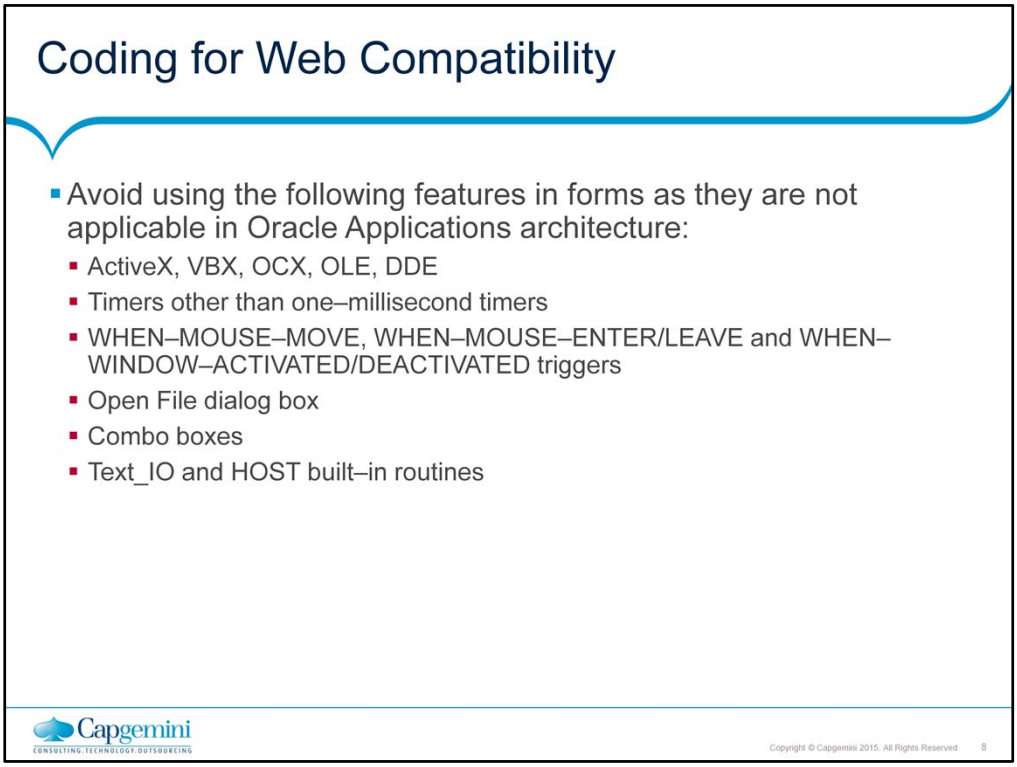


Performance

Performance is a critical issue in any application. Applications must avoid overloading the network that connects desktop client, application server, and database servers, since often it is network performance that most influences users' perceptions of application performance.

Oracle Applications are designed to minimize network traffic on all tiers. For example, they try to limit network round trips to one per user–distinguishable event by employing the following coding standards:

- Use database stored procedures when extensive SQL is required
- Code all non–SQL logic on the client side where possible
- Cache data on the client side where practical
- Base blocks on views that denormalize foreign key information where practical



Coding for Web Compatibility

Following Oracle Applications standards carefully will help ensure that your forms can be deployed on the Web.

You should avoid using the following features in your forms, as they are not applicable in this architecture:

ActiveX, VBX, OCX, OLE, DDE (Microsoft Windows–specific features that would not be available for a browser running on a Macintosh, for example, and cannot be displayed to users from within the browser)

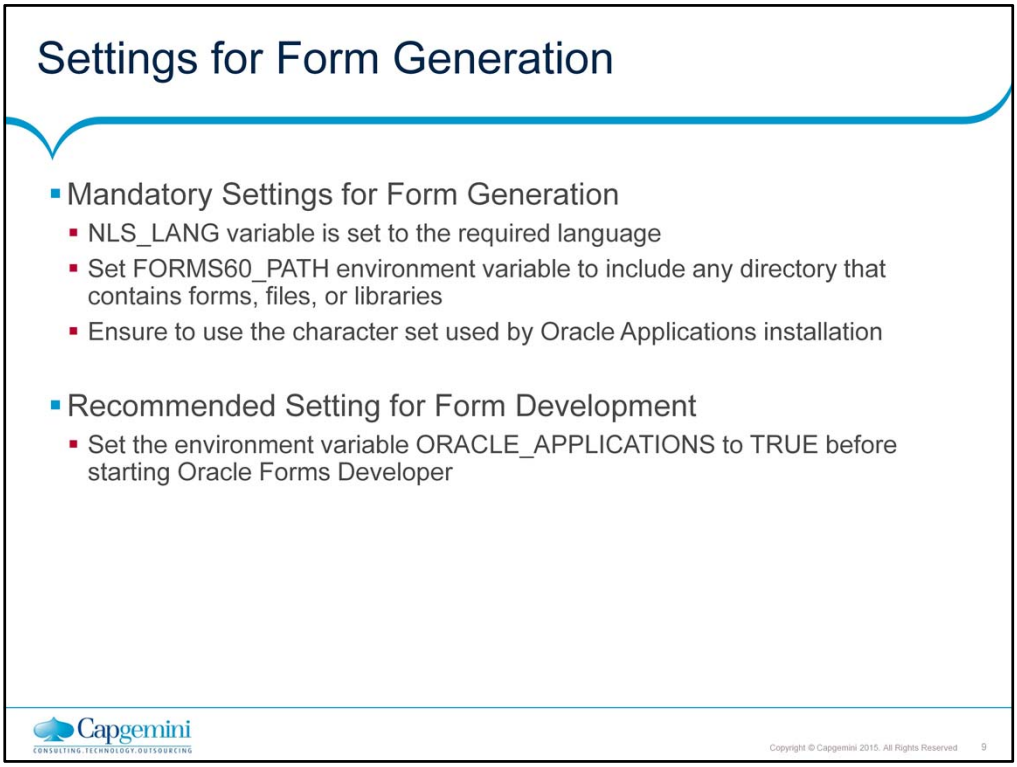
Timers other than one–millisecond timers (one–millisecond timers are treated as timers that fire immediately)

WHEN–MOUSE–MOVE, WHEN–MOUSE–ENTER/LEAVE and WHEN–WINDOW–ACTIVATED/DEACTIVATED triggers

Open File dialog box – It would open a file on the applications server, rather than on the client machine (where the browser is) as a user might expect

Combo boxes – Our standards do not use combo boxes anyhow

Text_IO and HOST built–in routines – These would take place on the applications server, rather than on the client machine (where the browser is) as a user might expect



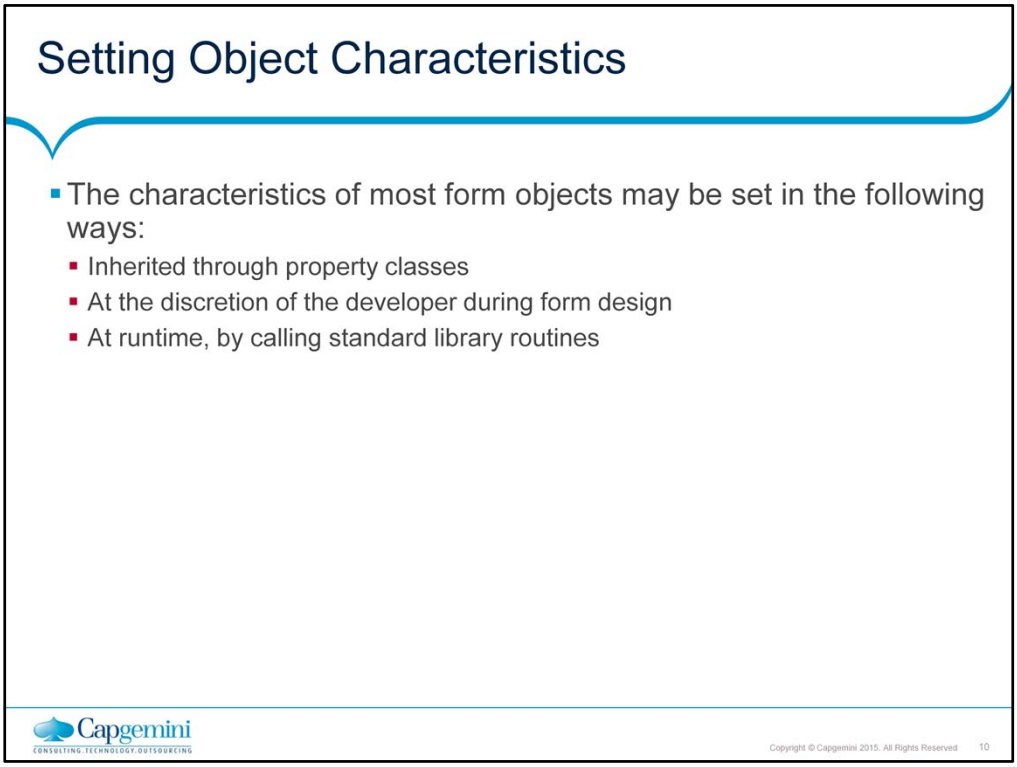
Mandatory Settings for Form Generation

At form generation time, make sure that NLS_LANG variable is set to the required language in the Windows NT registry or environment file (for Unix). Also ensure that the character set specified is the character set being used for Oracle Applications installation.

You must also set the value of FORMS60_PATH environment variable in environment file (Windows registry) to include any directory that contains forms, files, or libraries to be used to develop and generate forms. Specifically, include a path to the <$AU_TOP>/forms/US directory to be able to find all referenced forms, and a path to the <$AU_TOP>/resource directory to be able to find the Oracle Applications library files you need.

Recommended Setting for Form Development

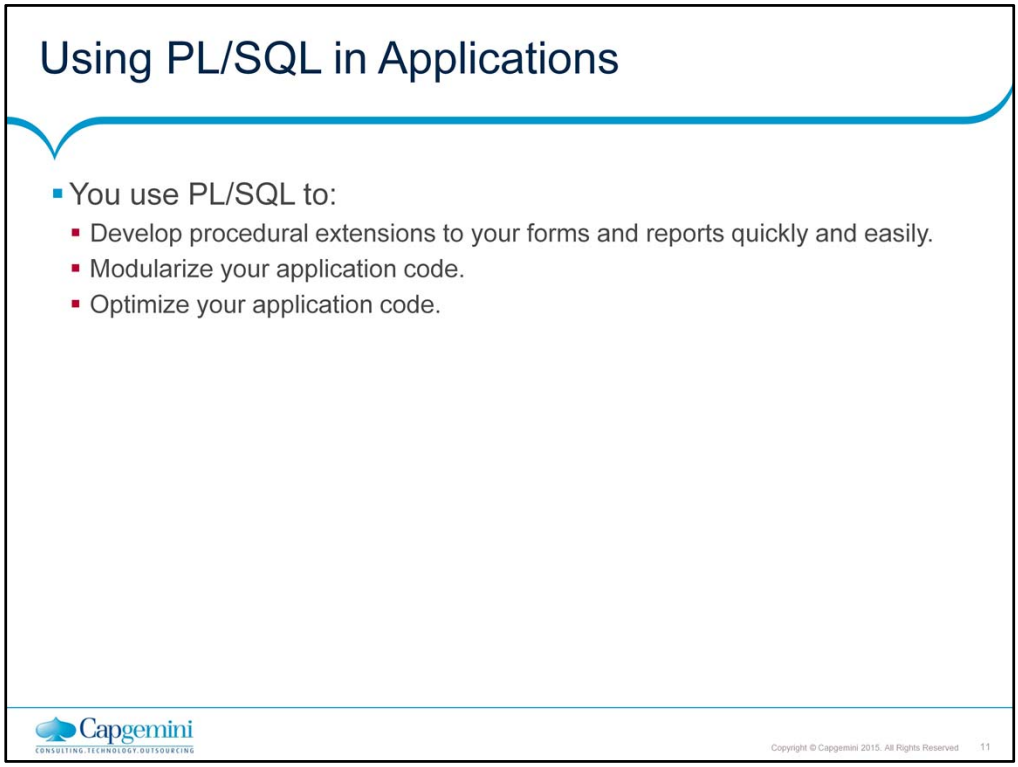
Oracle Forms Developer allows referenced objects to be overridden in the local form. Oracle Forms Developer also does not normally provide any indication that an object is referenced unless you set a special environment variable. Set the environment variable ORACLE_APPLICATIONS to TRUE before starting Oracle Forms Developer. This setting allows you to see the reference markers (little flags with an ”R” in them) on referenced objects so you can avoid changing referenced objects unintentionally. Any object referenced from the APPSTAND form must never be changed.



Setting Object Characteristics

The characteristics of most form objects, including modules, windows, canvases, blocks, regions, and items may be set in the following ways:

- Inherited through property classes, which cause certain properties to be identical in all forms (such as canvas visual attributes)
- At the discretion of the developer during form design (such as window sizes)
- At runtime, by calling standard library routines (such as window positions)



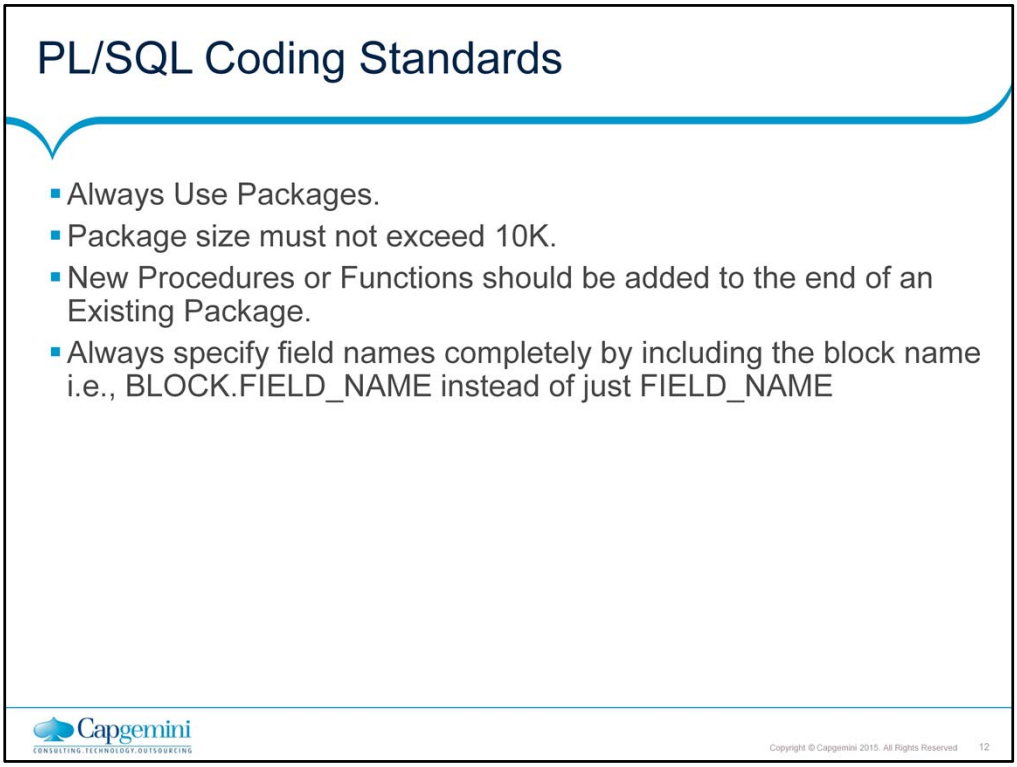
You can use PL/SQL procedures as part of an application that you build around Oracle Applications. By following the coding standards, you can create a PL/SQL procedure that integrates seamlessly with your application and with Oracle Applications.

You use PL/SQL to:

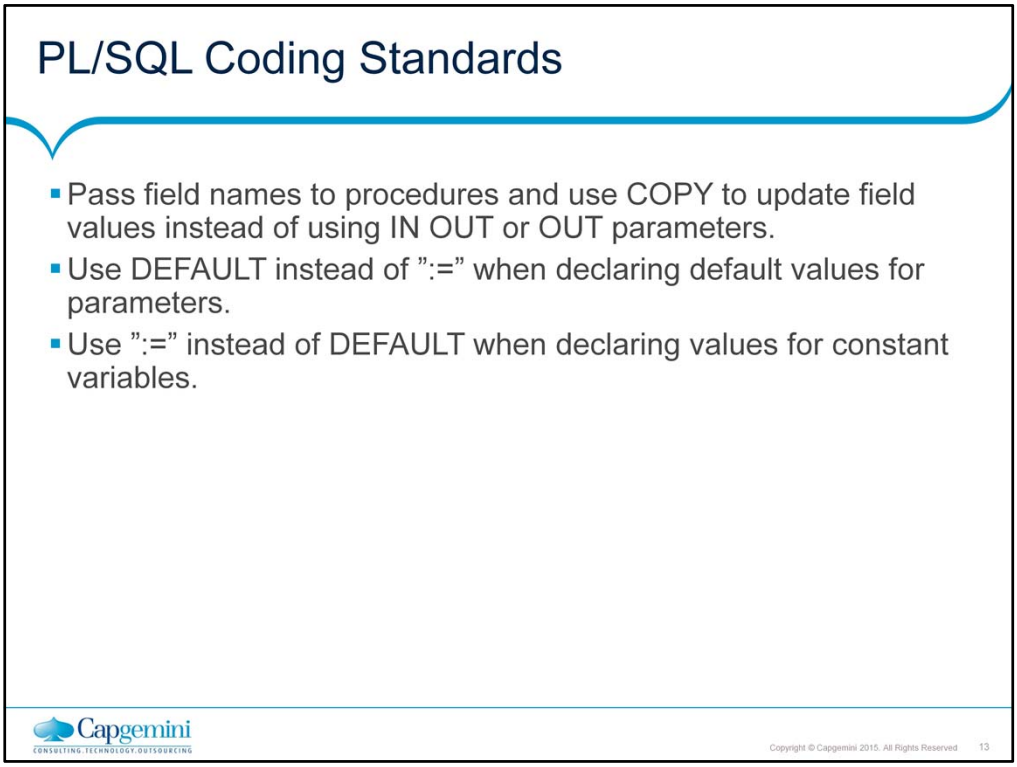
- Develop procedural extensions to your forms and reports quickly and easily
- Modularize your application code to speed development and improve maintainability
- Optimize your application code to reduce network traffic and improve overall performance

You can use PL/SQL to develop procedural extensions to custom forms and reports you create with Oracle tools. For example, to develop a form that follows Oracle Applications standards, organize form code into PL/SQL business rule procedures, item handlers, event handlers, and table handlers. Put very little PL/SQL code directly into form triggers because those triggers do not represent a logical model; they are simply event points that Oracle Forms provides for invoking procedural code. If you put most of the code in packaged PL/SQL procedures, and then call those procedures from triggers, you will have modular form code that is easy to develop and maintain.

You can also use PL/SQL to develop concurrent programs or stored procedures that are called from concurrent programs.



Always Use Packages
PL/SQL procedures should always be defined within packages. Create a package for each block of a form, or other logical grouping of code. A client–side (Oracle Forms) PL/SQL program unit's (package specification or body or stand–alone procedure.) source code and compiled code together must be less than 64K. This implies that the source code for a program unit cannot exceed 10K.
Adding New Procedures to Existing Packages
When you add new procedures or functions to existing packages (either stored in the database or in Oracle Forms libraries), you should usually add them to the end of the package (and package specification). If you add new procedures to the middle of the package specification and package, you must regenerate every form that references the package, or those forms may get ORA–4062 errors.
Using Field Names in Client–Side PL/SQL Packages
Always specify field names completely by including the block name i.e., BLOCK.FIELD_NAME instead of just FIELD_NAME. Just specifying the field name makes Oracle Forms to scan through the entire list of fields for each block in the form to locate the specified field and check if its name is ambiguous, potentially degrading your form performance. If the block name is included, Oracle Forms searches only the fields in that block and stops when it finds a match. Moreover, if you ever add more blocks, existing code continues to work since you specified field names unambiguously.



Field Names in Procedure Parameters

Pass field names to procedures and use COPY to update field values instead of using IN OUT or OUT parameters. This method prevents a field from being marked as changed whether or not you actually modify it in your procedure. Any parameter declared as OUT is always written to when the procedure exits normally.

For example, declare a procedure as

```
test(my_var VARCHAR2 IN)
```

and call it as test('block.field') instead of declaring the procedure as

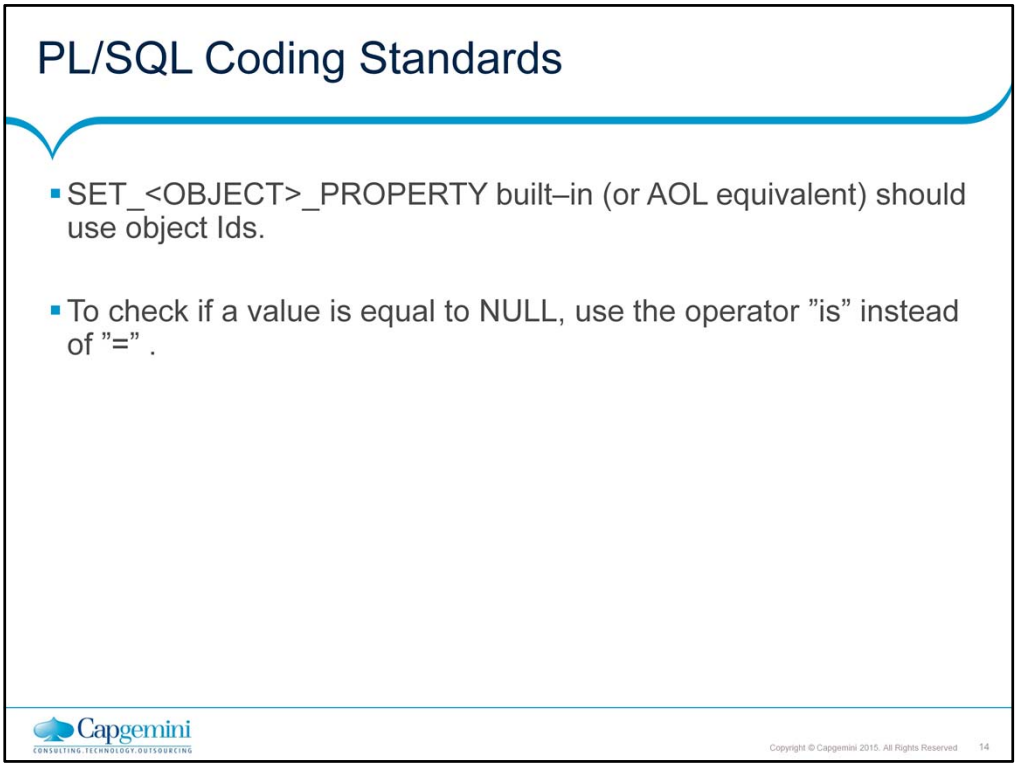
```
test(my_var VARCHAR2 IN OUT)
```

and calling it as test(:block.field).

Using DEFAULT

Use DEFAULT instead of ":=" when declaring default values for your parameters. DEFAULT is more precise because you are defaulting the values; the calling procedure can override the values.

Conversely, use ":=" instead of DEFAULT when declaring values for your constant variables. Using ":=" is more precise because you are assigning the values, not defaulting them; the values cannot be overridden.



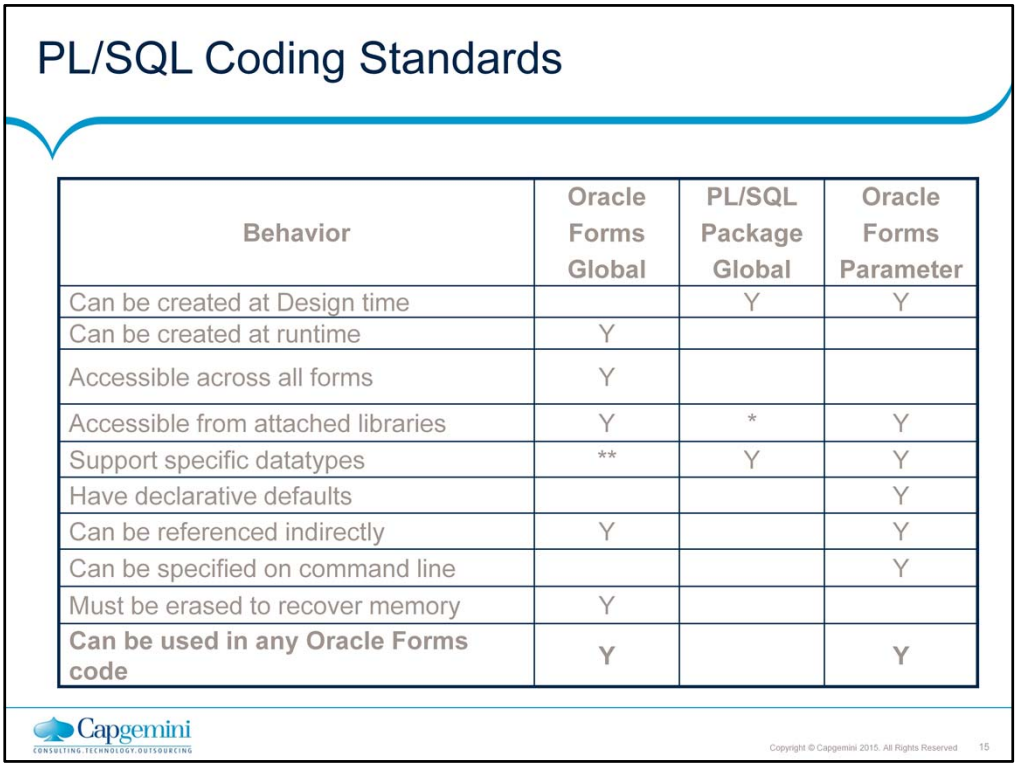
Use Object IDs

Any code that changes multiple properties of an object using the SET_<OBJECT>_PROPERTY built–in (or the Oracle Application Object Library equivalent) should use object IDs. First use the appropriate FIND_<OBJECT> built–in to get the ID, then pass the ID to the SET_<OBJECT>_PROPERTY built–in. Consider storing the ID in a package global so that its retrieved only once while the form is running.

Handling NULL Value Equivalence

handle NULL values very carefully in PL/SQL. For example, if a := NULL and b := NULL, the expression (a = b) evaluates to FALSE. In any ”=” expression where one of the terms is NULL, the whole expression will resolve to FALSE.

For this reason, to check if a value is equal to NULL, you must use the operator ”is” instead. If you’re comparing two values where either of the values could be equal to NULL, you should write the expression like this: ((a = b) or ((a is null) and (b is null))



Behavior	Oracle Forms Global	PL/SQL Package Global	Oracle Forms Parameter
Can be created at Design time		Y	Y
Can be created at runtime	Y		
Accessible across all forms	Y		
Accessible from attached libraries	Y	*	Y
Support specific datatypes	**	Y	Y
Have declarative defaults			Y
Can be referenced indirectly	Y		Y
Can be specified on command line			Y
Must be erased to recover memory	Y		
Can be used in any Oracle Forms code	**Y**		**Y**

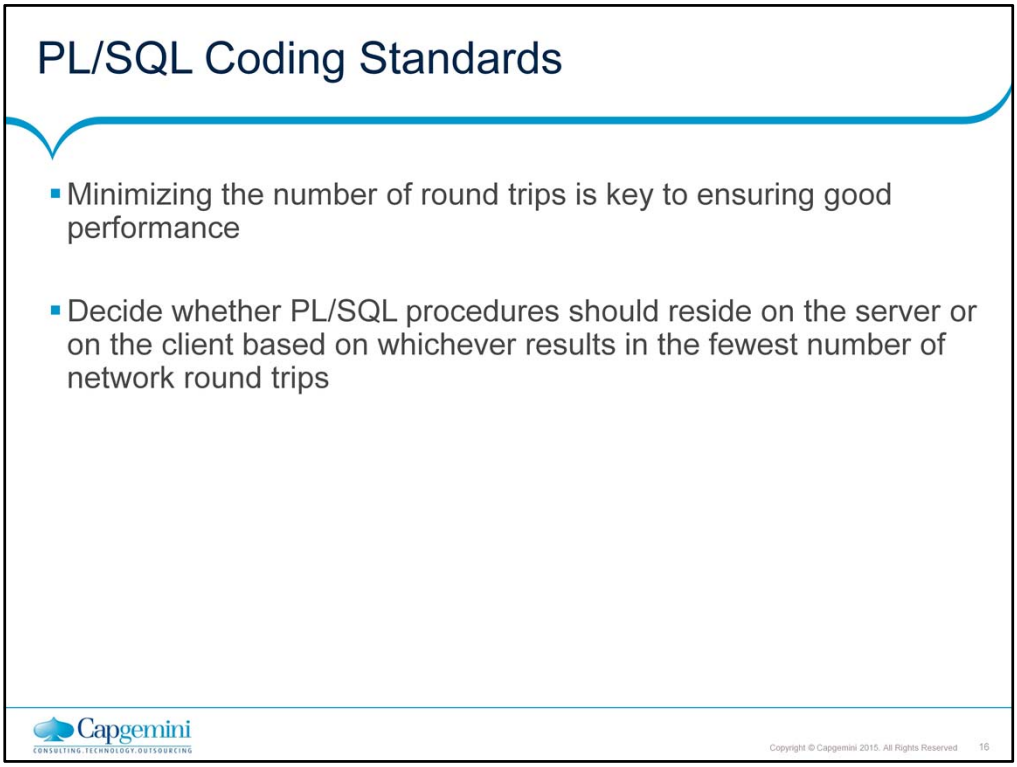
Global Variables

Oracle Forms Developer and PL/SQL support different types of global variables:

Oracle Forms Global: a variable in the ”global” pseudo–block of a form
PL/SQL Package Global: a global defined in the specification of a package
Oracle Forms Parameter: a variable created within the Oracle Forms Designer as a Parameter

The above table lists the characteristics of each type of variable, and enables you to select the type most appropriate for your code.

* A package variable defined in a form is not visible to any attached library; a variable defined in an attached library is visible to the form. (An Oracle Forms Global is visible to an attached library)
** Always CHAR(255)



Database Server Side versus Client Side

Performance is a critical aspect of any application. Because network round trips are very costly in a typical internet environment, minimizing the number of round trips is key to ensuring good performance.

You should decide whether your PL/SQL procedures reside on the server or on the client based on whichever results in the fewest number of network round trips. Here are some guidelines:

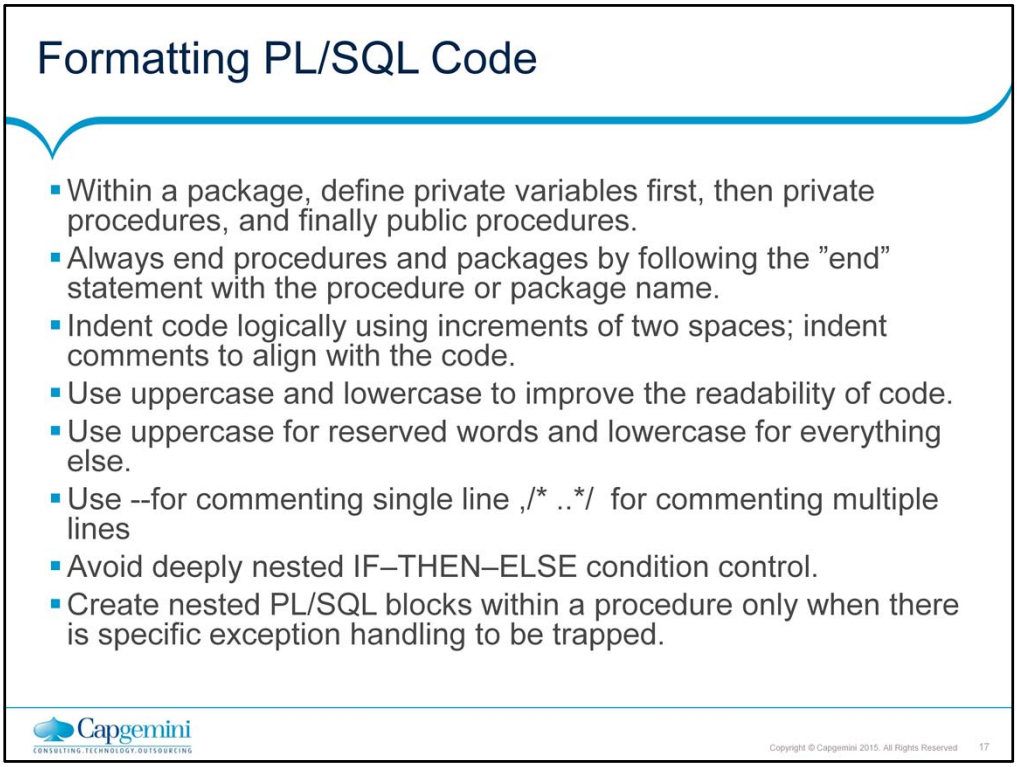
Procedures that call Oracle Forms built–ins (more generally, client built–ins) must reside on the client

Procedures that reference fields directly, either as :block.field or via NAME_IN/COPY, must reside on the client. You can avoid referencing fields directly by accepting field values or names as parameters to your PL/SQL procedures, which also improves your code's modularity

If a procedure contains three or more SQL statements, or becomes very complicated, the procedure usually belongs on the server

Procedures that perform no SQL and that need no database access should reside wherever they are needed

If a procedure is called from the server, it must reside on the server. If a procedure is called from both client and server, it should be defined in both places, unless the procedure is very complicated and double maintenance is too costly. In the latter case, the procedure should reside on the server.



Formatting PL/SQL Code

This section contains recommendations for formatting PL/SQL code.

Within a package, define private variables first, then private procedures, and finally public procedures.

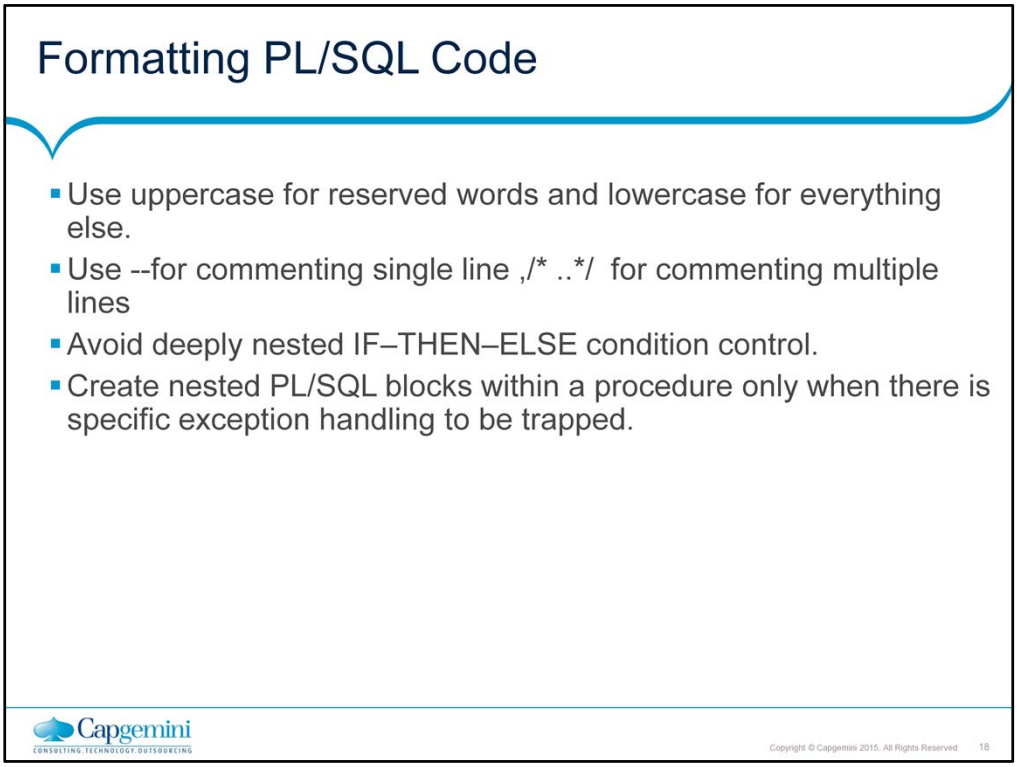
Always end procedures and packages by following the ”end” statement with the procedure or package name to help delineate procedures.

Indent code logically. Using increments of two spaces provides an easy way to track your nested cases.

Indent SQL statements as follows:

Example

```
DECLARE
  CURSOR employees IS
    SELECT empno
    FROM emp
    WHERE deptno = 10
        AND ename IN ('WASHINGTON', 'MONROE')
        AND mgr = 2701;
```



Use ”– –” to start comments so that you can easily comment out large portions of code during debugging with ”/* ... */”
Indent comments to align with the code being commented
When commenting out code, start the comment delimiter in the leftmost column. When the code is clearly no longer needed, remove it entirely
Use uppercase and lowercase to improve the readability of your code (PL/SQL is case–insensitive). As a guideline, use uppercase for reserved words and lowercase for everything else
Avoid deeply nested IF–THEN–ELSE condition control. Use IF–THEN–ELSIF instead

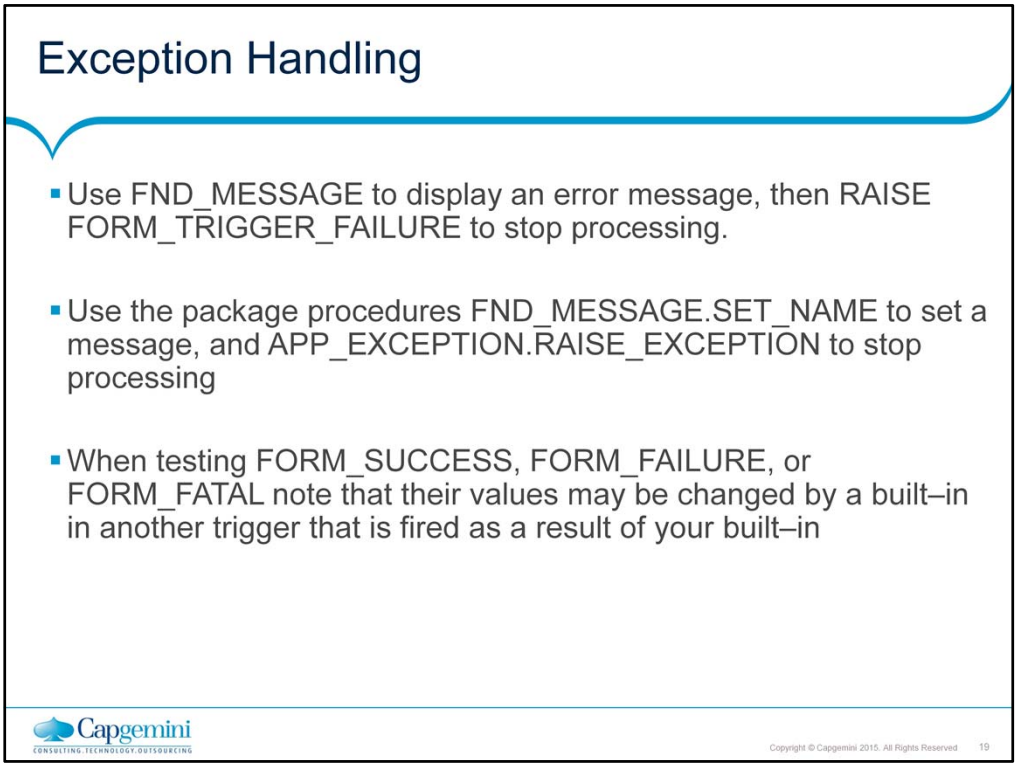
Example of Bad Style

```
IF ... THEN ... ELSE
        IF ... THEN ... ELSE
                IF ... THEN ... ELSE
                END IF
        END IF
END IF;
```

Example of Good Style

Only create nested PL/SQL blocks (BEGIN/END pairs) within a procedure when there is specific exception handling you need to trap

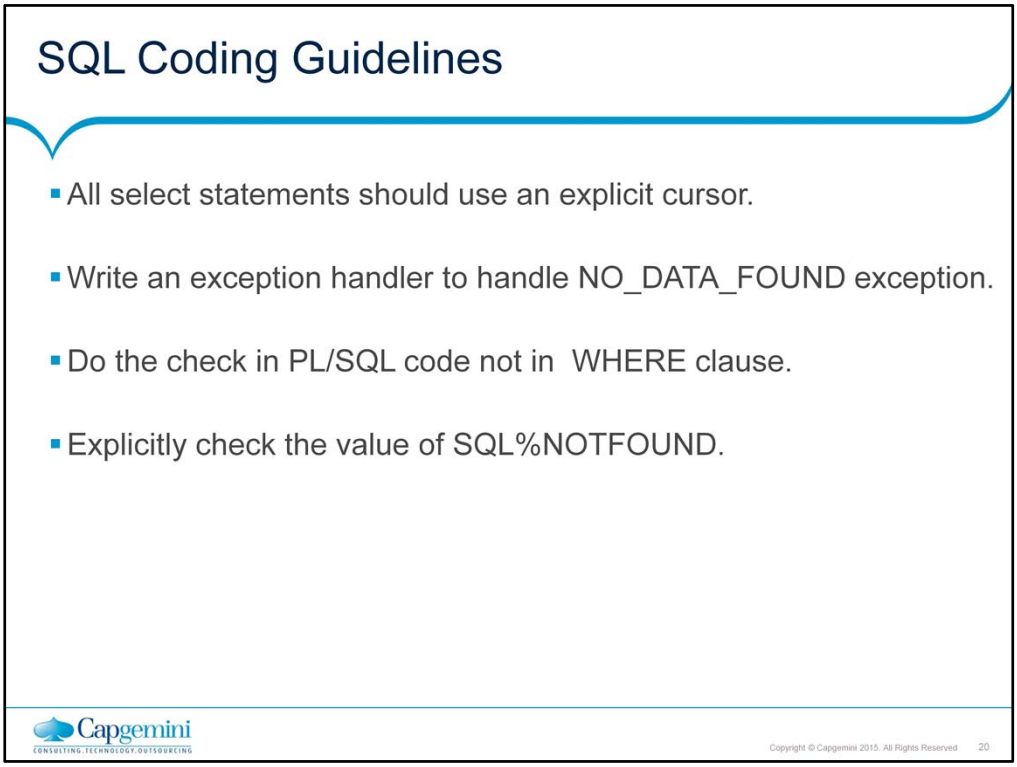
```
IF ... THEN ...
ELSIF ... THEN ...
ELSIF ... THEN ...
ELSIF ... THEN ...
ELSE ...
END IF;
```



Errors in Oracle Forms PL/SQL
If a failure occurs in Oracle Forms PL/SQL and you want to stop further processing, use FND_MESSAGE to display an error message, then RAISE FORM_TRIGGER_FAILURE to stop processing:

```
IF (error_condition) THEN
  fnd_message.set_name(appl_short_name,
message_name);
  fnd_message.error;
  RAISE FORM_TRIGGER_FAILURE;
END IF;
```

Note: RAISE FORM_TRIGGER_FAILURE causes processing to stop quietly without error notification, you must display any messages yourself using FND_MESSAGE before raising the exception.



Follow these guidelines for all SQL that you code:

Use ”select from DUAL” instead of ”select from SYS.DUAL”. Do not use SYSTEM.DUAL

All SELECT statements should use an explicit cursor. Implicit SELECT statements actually cause 2 fetches to execute: one to get the data, and one to check for the TOO_MANY_ROWS exception. Avoid this by FETCHing just a single record from an explicit cursor

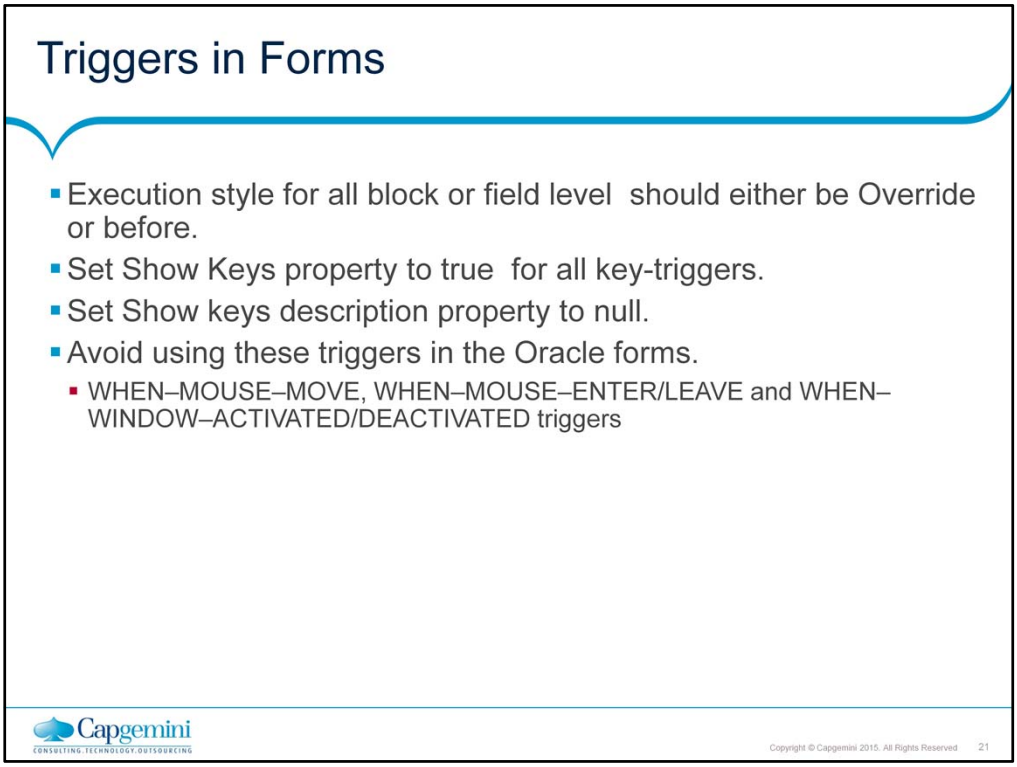
To SELECT into a procedure parameter, declare the parameter as IN OUT, whether or not the parameter value is referenced, unless the parameter is a field

A single–row SELECT that returns no rows raises the exception NO_DATA_FOUND. An INSERT, UPDATE, or DELETE that affects no rows does not raise an exception. Explicitly check the value of SQL%NOTFOUND if no rows is an error.

To handle NO_DATA_FOUND exceptions, write an exception handler. Do not code COUNT statements to detect the existence of rows unless that is your only concern.

When checking the value of a field or PL/SQL variable against a literal, do the check in PL/SQL code, not in a WHERE clause.

Do not check for errors due to database integrity problems. For example, if a correct database would have a table SYS.DUAL with exactly one row in it, you do not need to check if SYS.DUAL has zero or more than one row or if SYS.DUAL exists



Follow these general rules for triggers in the forms.

Execution Style

The 'Execution Style' for all block or field level triggers should either be Override or Before. In general, use style Before, since usually the form–level version of the trigger should also fire. The exception is if there is a flexfield call in the form–level POST–QUERY trigger, but you reset the query status of the block in the block level POST–QUERY. In that case, the block–level POST–QUERY should use Execution Style After.

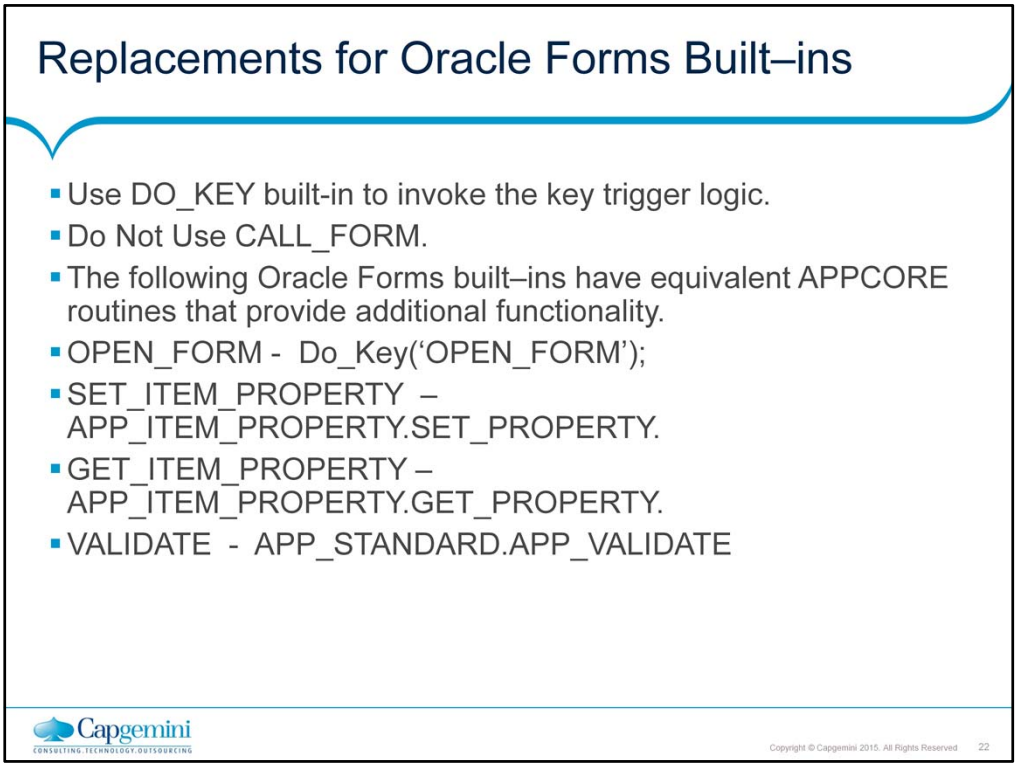
KEY– Trigger Properties

Set the "Show Keys" property to True for all KEY– triggers you code, except those that you are disabling (which should have "Show Keys" set to False). Always set the "Show Keys Description" property to NULL.

WHEN–CREATE–RECORD in Dynamic Query–Only Mode

The WHEN–CREATE–RECORD trigger fires even when the block does not allow inserts. Check if the block allows insert if you have logic in this trigger and your block may dynamically have insert–allowed "FALSE":

```
IF GET_ITEM_PROPERTY('<BLOCK>',
INSERT_ALLOWED) = FALSE THEN
  null;
ELSE
  <your logic here>;
END IF;
```



These standards require that certain built–ins be avoided entirely, or ”wrapper” routines be called in their place. For many built–ins, there are multiple methods of invocation. Call the built–in directly, to give the standard forms behavior. For some built–ins, there are standard Oracle Applications behaviors, which are invoked by calling APP_STANDARD.EVENT.
Many of these built–ins have a key and a KEY– trigger associated with them. If there is any additional logic which has been added to the KEY–trigger that you want to take advantage of, you can invoke the trigger by using the DO_KEY built–in. This is the same result you would get if the user pressed the associated key.

Do not use CALL_FORM Oracle Forms built–in.
This built–in is incompatible with OPEN_FORM, which is used by Oracle Applications routines. Use FND_FUNCTION.EXECUTE instead of either CALL_FORM or OPEN_FORM whenever a form to be opened programmatically. Using FND_FUNCTION.EXECUTE allow the form to open without bypassing Oracle Applications security, and takes care of finding the correct directory path for the form.

Review Question

- Which all are the data types to be avoided?
- What are views? Why is an LOV view simpler
- than a block view?
- Give some advantages of views.
- What are WHO columns?

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

ERP – Oracle Apps

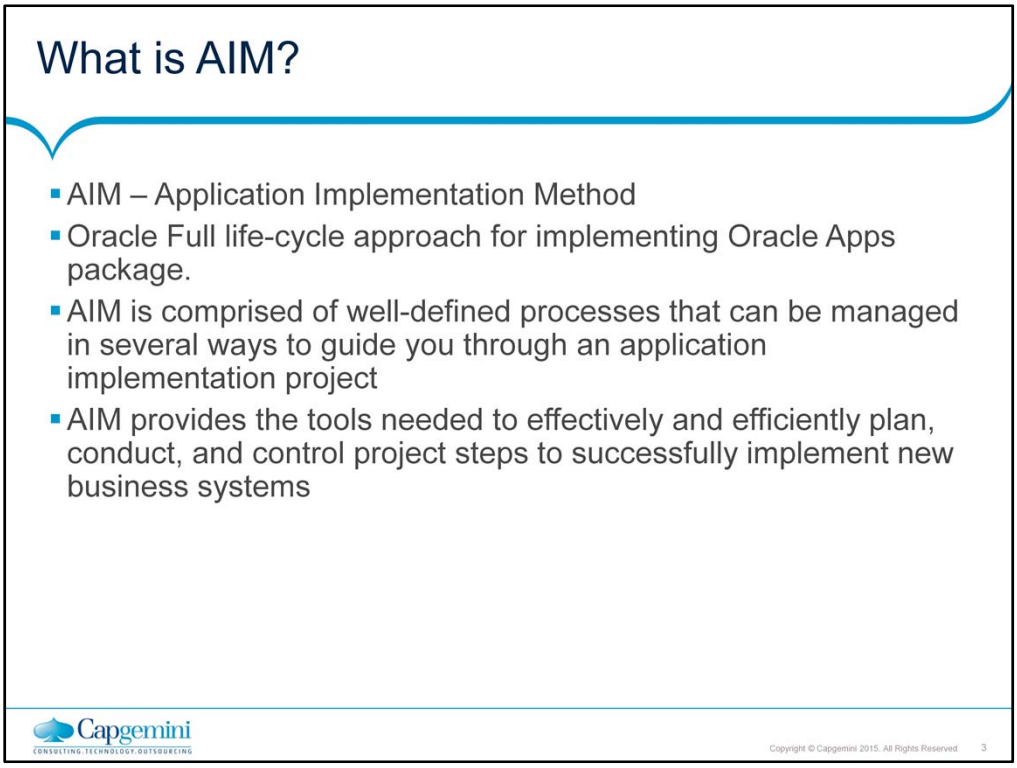
Lesson 17: Best Practices - Aim Methodology

Lesson Objectives

- What is AIM?
- Activities Supported – AIM
- AIM - Phases
- AIM – Data Conversion (CV) Methodology
- Data Conversion (CV) – Objective & Steps
- Documentations – Data Conversions
- Data Conversion – Processes
- Usage & Deliverables – Each Conversion Documents
- Customization Process - Documents

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2



Oracle's Application Implementation Method (AIM) is a proven approach for implementing Oracle Applications.

AIM is comprised of well-defined processes that can be managed in several ways to guide you through an application implementation project. AIM provides the tools needed to effectively and efficiently plan, conduct, and control project steps to successfully implement new business systems.

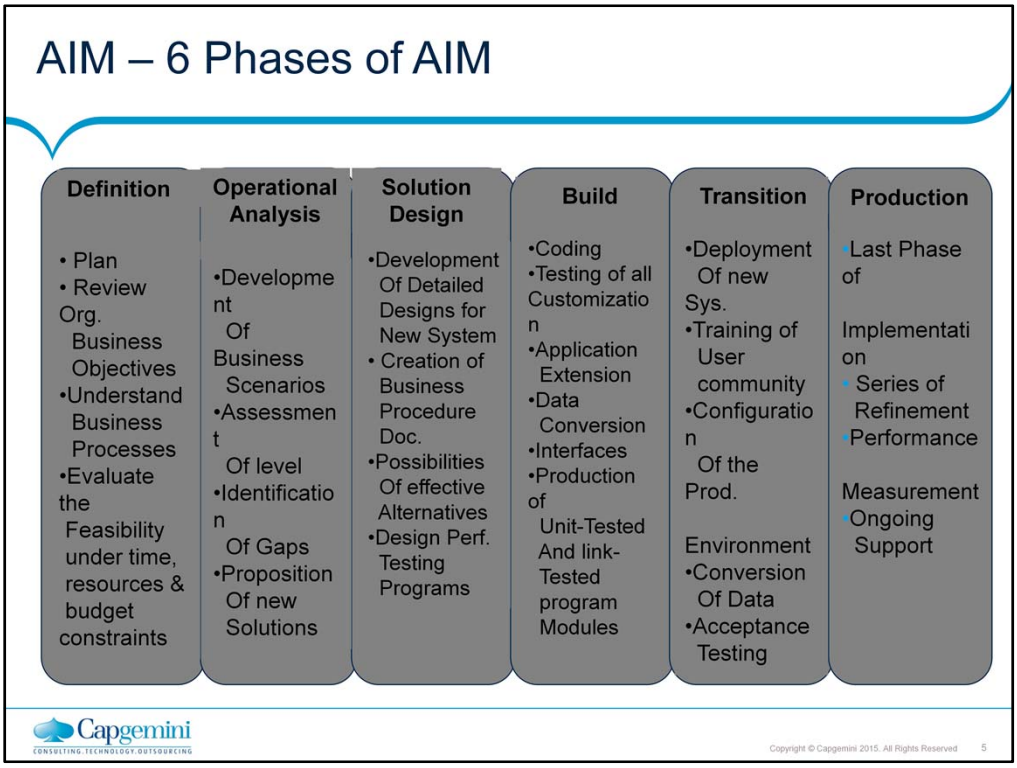
AIM is very broad in its support of all of the activities within your implementation project including:
.Planning
.requirements definition
. business process alignment and modeling
. customization
. interfaces and integration between systems
. data conversion
. organization change management including specific support for executive, management, and user groups
. application and technical architecture including network and server design
. reporting and information access systems
. security and access control

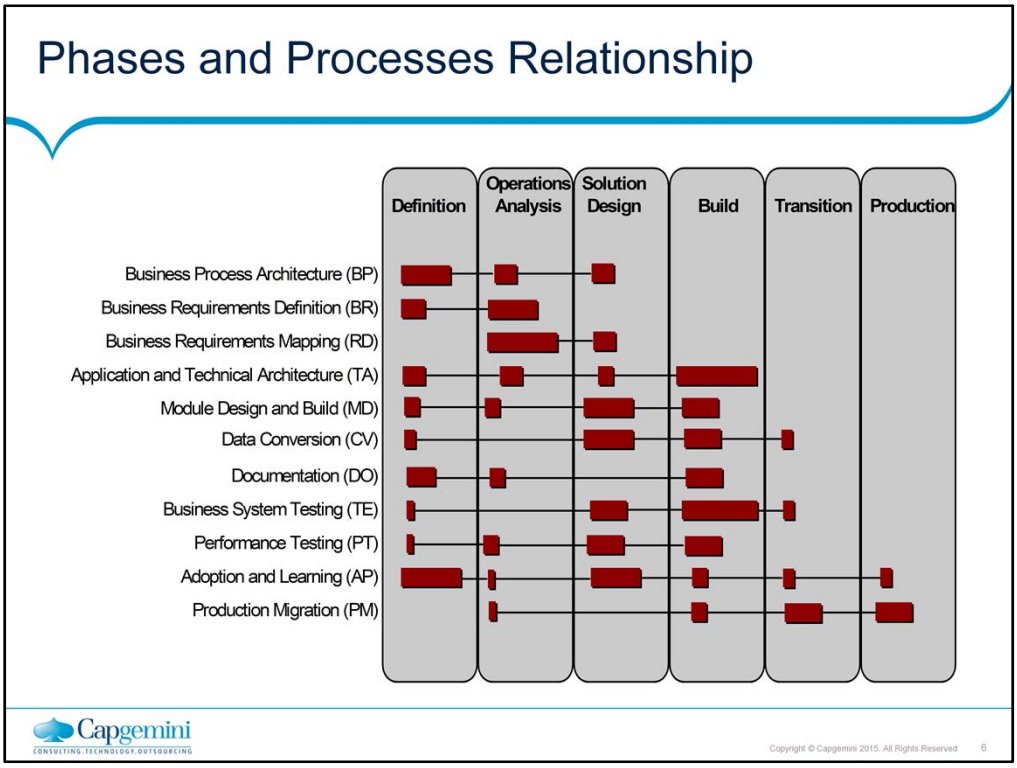
AIM Processes

- Planning
- Requirement Definition
- Business Process alignment and modeling
- Customization
- Interfaces and integration between systems
- Data Conversion
- Organization Change Management
- Application and technical architecture
- Reporting and information access systems
- Security and access control

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4


AIM – 6 Phases of AIM
Definition
• Plan
• Review Org. Business Objectives
•Understand Business Processes
•Evaluate the Feasibility under time, resources & budget constraints
Operational Analysis
•Development Of Business Scenarios
•Assessment Of level
•Identification Of Gaps
•Proposition Of new Solutions
Solution Design
•Development Of Detailed Designs for New System
• Creation of Business Procedure Doc.
•Possibilities Of effective Alternatives
•Design Perf. Testing Programs
Build
•Coding
•Testing of all Customization
•Application Extension
•Data Conversion
•Interfaces
•Production of Unit-Tested And link-Tested program Modules
Transition
•Deployment Of new Sys.
•Training of User community
•Configuration Of the Prod.
Environment
•Conversion Of Data
•Acceptance Testing
Production
•Last Phase of
Implementation
• Series of Refinement
•Performance
Measurement
•Ongoing Support
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved 5

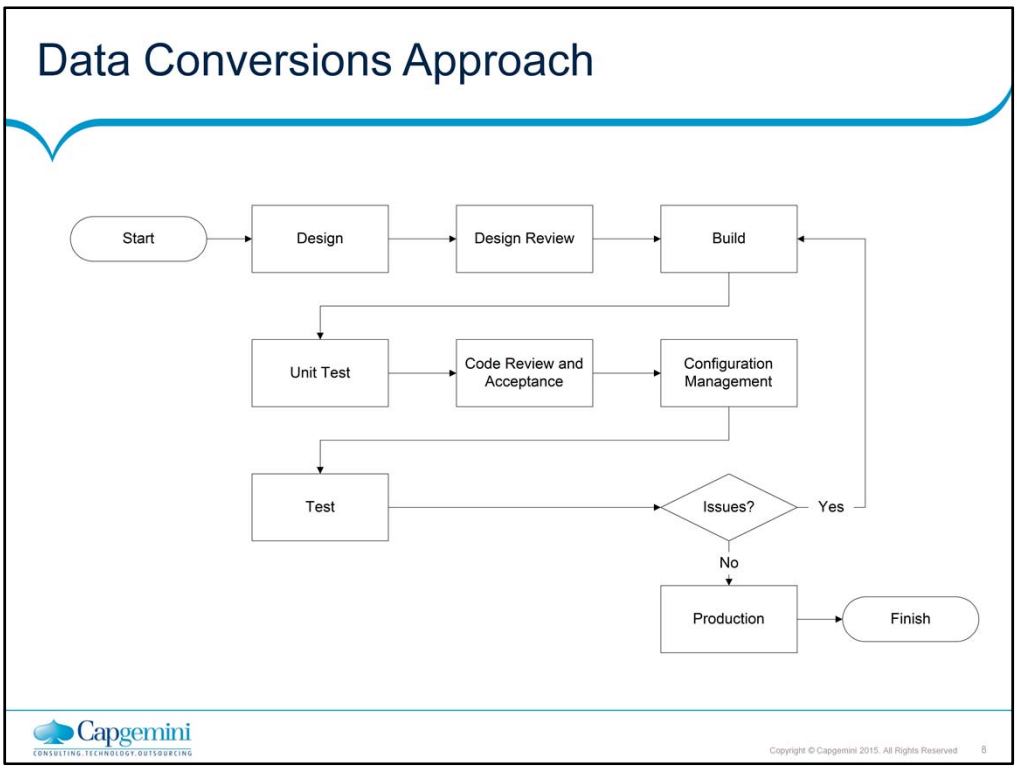

Phases and Processes Relationship
Definition
Operations Analysis
Solution Design
Build
Transition
Production
Business Process Architecture (BP)
Business Requirements Definition (BR)
Business Requirements Mapping (RD)
Application and Technical Architecture (TA)
Module Design and Build (MD)
Data Conversion (CV)
Documentation (DO)
Business System Testing (TE)
Performance Testing (PT)
Adoption and Learning (AP)
Production Migration (PM)
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
6

Data Conversions Approach

- Oracle Conversion Projects mainly involves migration of data from Legacy system to Oracle Applications or from lower version of Ora Apps to higher version. This diagram describes the project execution model.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

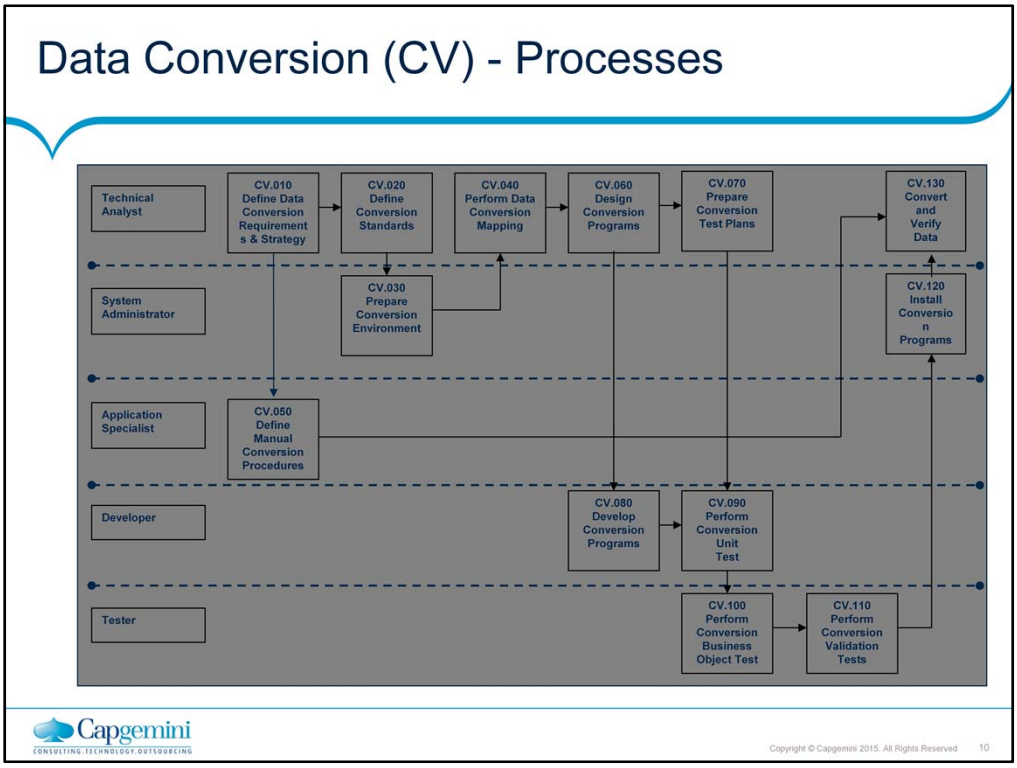

Data Conversions Approach
Start
Design
Design Review
Build
Unit Test
Code Review and Acceptance
Configuration Management
Test
Issues?
Yes
No
Production
Finish
Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved 8

Documentations – Data Conversion

- CV.010 - Define Data Conversion Requirements and Strategy
- CV.020 - Define Conversion Standards
- CV.030 - Prepare Conversion Environment
- CV.040 - Perform Conversion Data Mapping
- CV.050 - Define Manual Conversion Procedures
- CV.060 - Design Conversion Programs
- CV.070 - Prepare Conversion Test Plans
- CV.080 - Develop Conversion Programs
- CV.090 - Perform Conversion Unit Tests
- CV.100 - Perform Conversion Business Object Tests
- CV.110 - Perform Conversion Validation Tests
- CV.120 - Install Conversion Programs
- CV.130 - Convert and Verify Data

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 9


Data Conversion (CV) - Processes
Technical Analyst
CV.010 Define Data Conversion Requirements & Strategy
CV.020 Define Conversion Standards
CV.040 Perform Data Conversion Mapping
CV.060 Design Conversion Programs
CV.070 Prepare Conversion Test Plans
CV.130 Convert and Verify Data
System Administrator
CV.030 Prepare Conversion Environment
CV.120 Install Conversion Programs
Application Specialist
CV.050 Define Manual Conversion Procedures
Developer
CV.080 Develop Conversion Programs
CV.090 Perform Conversion Unit Test
Tester
CV.100 Perform Conversion Business Object Test
CV.110 Perform Conversion Validation Tests
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
10

Terms Used for the Key Responsibilities

- Technical Analyst : Performs conversion mapping and design conversion modules
- System Administrator: Support the data conversion and provide assistance as needed for accessing the legacy systems.
- Application Specialist: Provides Knowledge and guidance regarding application functionality
- Developer: Code Conversion Modules. Develop Conversion unit test specifications and perform the conversion unit test. Perform the data conversion, if required.
- Tester: Help with conversion business object and validation testing to verify data correctness.
- Business Object: Physical or logical object of significance to a business

  e.g. vendors, customers, people, assignments etc.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

CV.010 – Define Data Conversion Requirements and Strategy

- Define the scope of the conversion project
- Conversion Objective and approach
- Prepare a strategy for converting information from the legacy systems to the new application environment
- The task provides a roadmap for performing the conversion of data from the legacy system to the new Oracle System
- Deliverable Data Conversion Requirements and Strategy

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

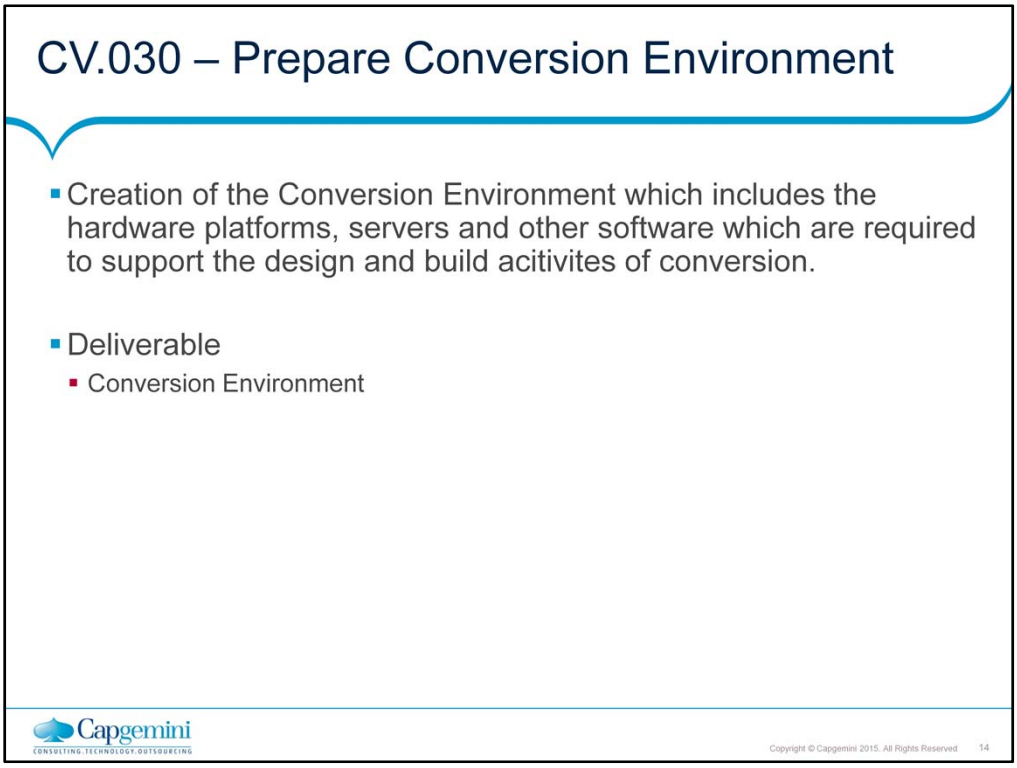
 12

CV.020 – Define Conversion Standards

- Creation of the Conversion Standards the Conversion Team needs to follow when performing conversion tasks.
- Documents the file structure and naming conventions for the legacy and target systems as well as for any automated tools which would help in facilitate the conversion process.
- Deliverable
  - Conversion Standards

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 13


CV.030 – Prepare Conversion Environment
Creation of the Conversion Environment which includes the hardware platforms, servers and other software which are required to support the design and build acitivites of conversion.
Deliverable
Conversion Environment
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
14

CV.040 – Perform Conversion Data Mapping

- Mapping the data elements from the legacy systems to the target Oracle Applications
- Identification – Data Sources and target tables and columns
- Identification – Data Defaults, Validations, processing, translation, filters and foreign key rules.

- Deliverable

  Conversion Data Mapping
  - provides the detailed data mapping and extract file layout for each legacy data elements one plans to convert to Oracle.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

CV.050 – Define Manual Conversion Procedures

- Define the plan to convert the business objects which require manual conversion.

- Deliverable

  Manual Conversion Procedures
  - Defines how to manually convert a business object
  - Discuss the impact if one does not successfully convert the business objects in the required time frame.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

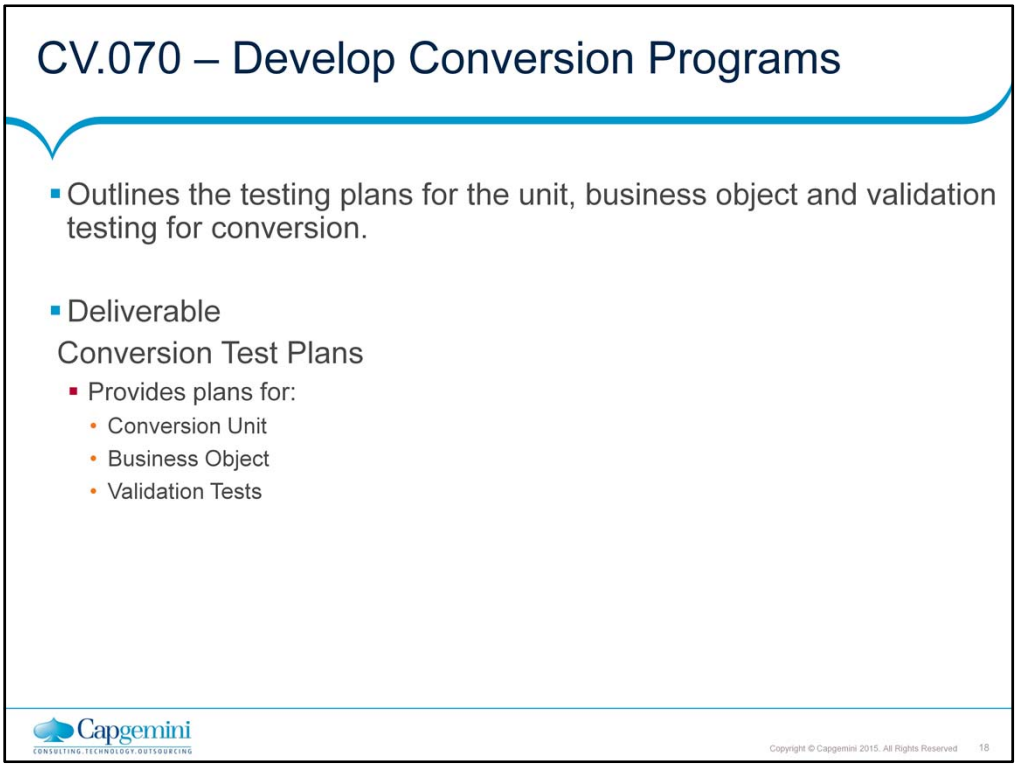
CV.060 – Define Data Conversion Requirements and Strategy

- Design and Document the Conversion Programs
- Deliverable

  Conversion Program Designs
  - Defines the key assumptions, rules and logic needed to create the conversion Modules.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17


CV.070 – Develop Conversion Programs
Outlines the testing plans for the unit, business object and validation testing for conversion.
Deliverable
Conversion Test Plans
Provides plans for:
Conversion Unit
Business Object
Validation Tests
Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
18

CV.080 – Develop Conversion Programs

- Creation of the conversion programs that performs all of the functions required to convert legacy business objects to the target Oracle Applications.
- Conversion Programs of each Business Object consist:
- Download Program
- Interface Table Creation Program
- Upload Program
- Translation Program
- Interface and Validation Program

- Deliverable

  Conversion Program

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19

CV.090 – Perform Conversion Unit Tests

- Purpose – to test the conversion programs to verify that all programs work without errors and according to the conversion testing specifications pre-defined in the CV.070.

- Deliverable

  Unit-Tested Conversion Programs.

  - It includes conversion program source code which has been tested to verify the processing logic of each program module functions without errors.

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 20

CV.100 – Perform Conversion Business Object Tests

- Testing the complete conversion of each business object by executing all conversion modules for the business object in the appropriate sequence and thus verifying so that the resulting data is correct.

- Deliverable

  Business Object-Tested Conversion Programs

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

CV.110 – Perform Conversion Validation Tests

- Validation of the target applications function correctly with the converted business objects

- Deliverable

  Validation-Tested Conversion Programs

  - It includes conversion programs that have been tested to verify that the resulting converted legacy data performs correctly in the entire suite of Oracle Applications.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22

CV.120 – Install Conversion Programs

- Perform and document the installation of the conversion programs in the production environment.
- Deliverable
  Installed Conversion Programs

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

CV.130 – Convert and Verify Data

- Conversion and Migration of the production data from the old system to the new Oracle Production Environment
- Completion of task -> data is ready for production use.

- Deliverable

  Converted and Verified Data

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

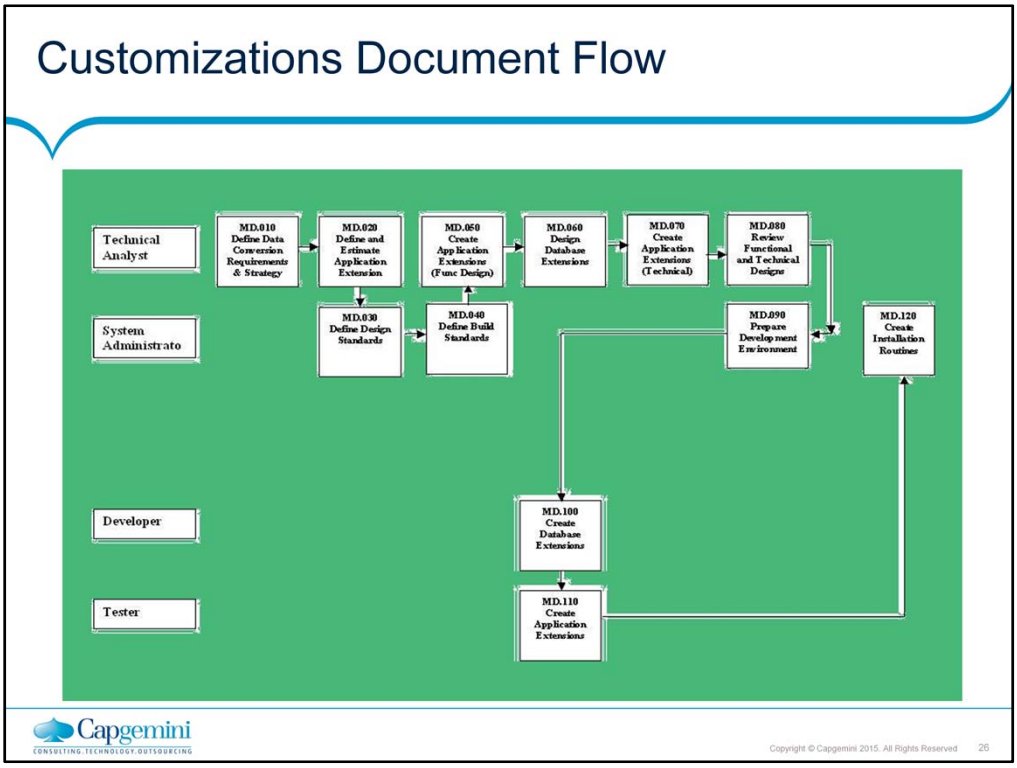
 24

Customization Documents

- MD.010 - Define Application Extension Strategy
- MD.020 - Define and Estimate Application Extensions
- MD.030 - Define Design Standards
- MD.040 - Define Build Standards
- MD.050 - Create Application Extensions Functional Design
- MD.060 - Design Database Extensions
- MD.070 - Create Application Extensions Technical Design
- MD.080 - Review Functional and Technical Designs
- MD.090 - Prepare Development Environment
- MD.100 - Create Database Extensions
- MD.110 - Create Application Extension Modules
- MD.120 - Create Installation Routines

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 25


Customizations Document Flow
Technical Analyst
System Administrato
Developer
Tester
MD.010 Define Data Conversion Requirements & Strategy
MD.020 Define and Estimate Application Extension
MD.030 Define Design Standards
MD.040 Define Build Standards
MD.050 Create Application Extensions (Func Design)
MD.060 Design Database Extensions
MD.070 Create Application Extensions (Technical)
MD.080 Review Functional and Technical Designs
MD.090 Prepare Development Environment
MD.100 Create Database Extensions
MD.110 Create Application Extensions
MD.120 Create Installation Routines
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
26

Summary

- In this lesson, you should have learned how to:
- What is AIM?
- Activities Supported – AIM
- AIM - Phases
- AIM – Data Conversion (CV) Methodology
- Data Conversion (CV) – Objective & Steps
- Documentations – Data Conversions
- Data Conversion – Processes
- Usage & Deliverables – Each Conversion Documents
- Customization Process - Documents

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 27

Summary

- In this lesson, you should have learned how to:
  - Basic of Coding Standards
  - Handlers
  - Triggers
  - SQL
  - PL/SQL Code

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 28

ERP- Oracle Apps

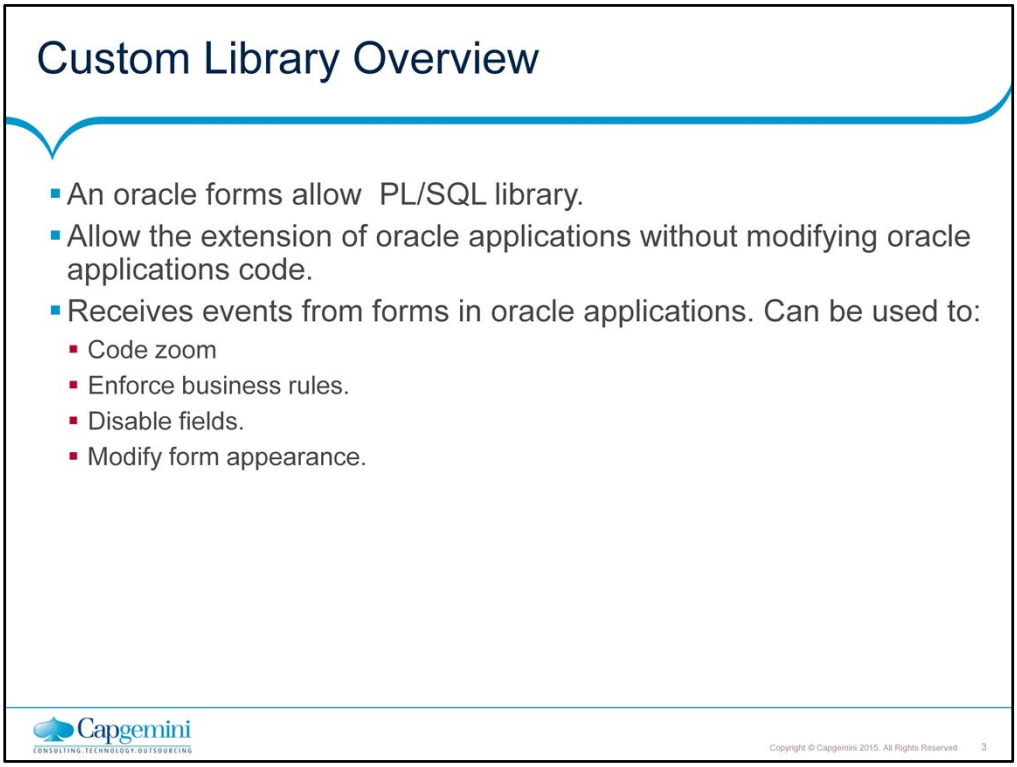
Lesson 18: Customization using Custom.pll

Lesson Objectives

- List the coverage for this lesson
- Custom Library Overview.
- Architecture of the CUSTOM Library.
- Routines in the CUSTOM library.
- Adding packages to the CUSTOM library.
- Coding Restrictions.
- Events passed to the CUSTOM library.
- Examples.
- Zoom LOV.
- Summary.
- Review Question

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

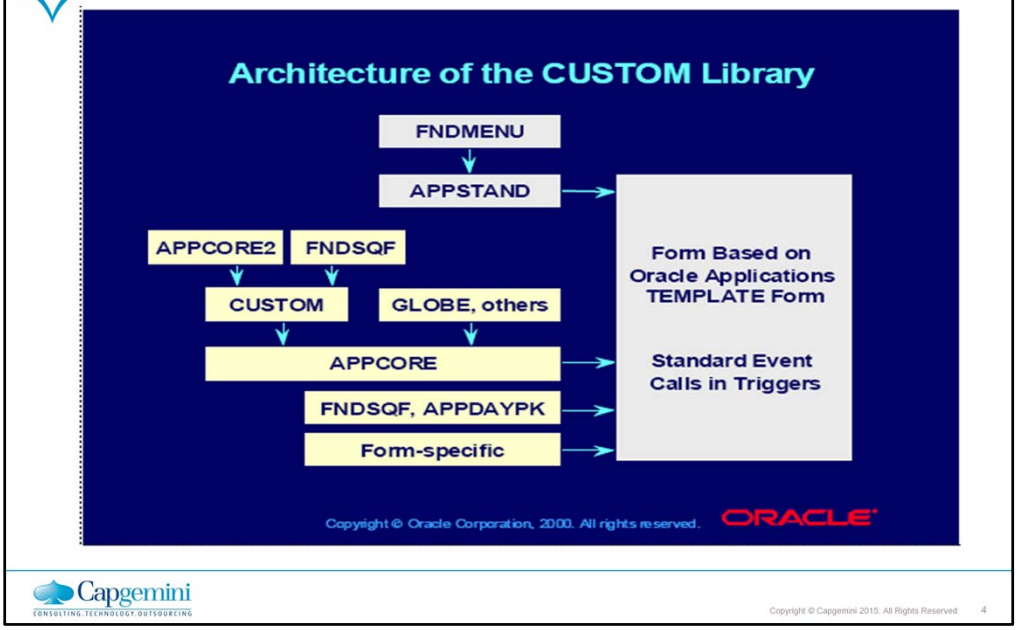


CUSTOM.pll is used to add extensions to Oracle's form Functionality. Some of the common scenarios where CUSTOM.pll can be used are:-

1. Enabling/Disabling the fields
2. Changing the List of Values in a LOV field at runtime
3. Defaulting values
4. Additional record level validations
5. Navigation to other screens
6. Enabling Special Menu

The CUSTOM.pll library is a standard Oracle Forms PL/SQL library that is supplied by Oracle with the Oracle
Applications. This is Oracle's built-in feature that allows the customer to enhance the standard functionality of the
Applications by implementing site-specific business rules. Every Oracle Forms - based eBusiness screen, and any
custom form developed using the Oracle Application development standards, will access the CUSTOM library.
This makes an ideal point of creating business rules that effect the entire organization. This is the only method of
forms enhancement whose functionality is supported by Oracle World Wide Support. Although any enhancements
coded by the customer are not directly supported by Oracle World Wide Support.

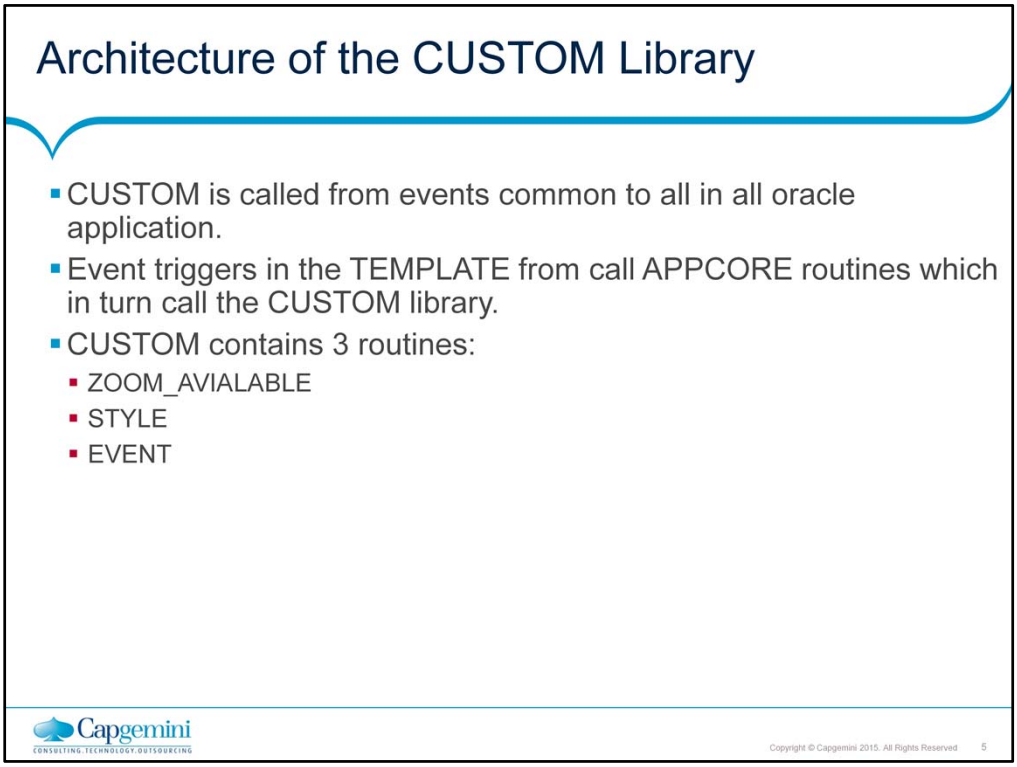
Architecture of the CUSTOM Library



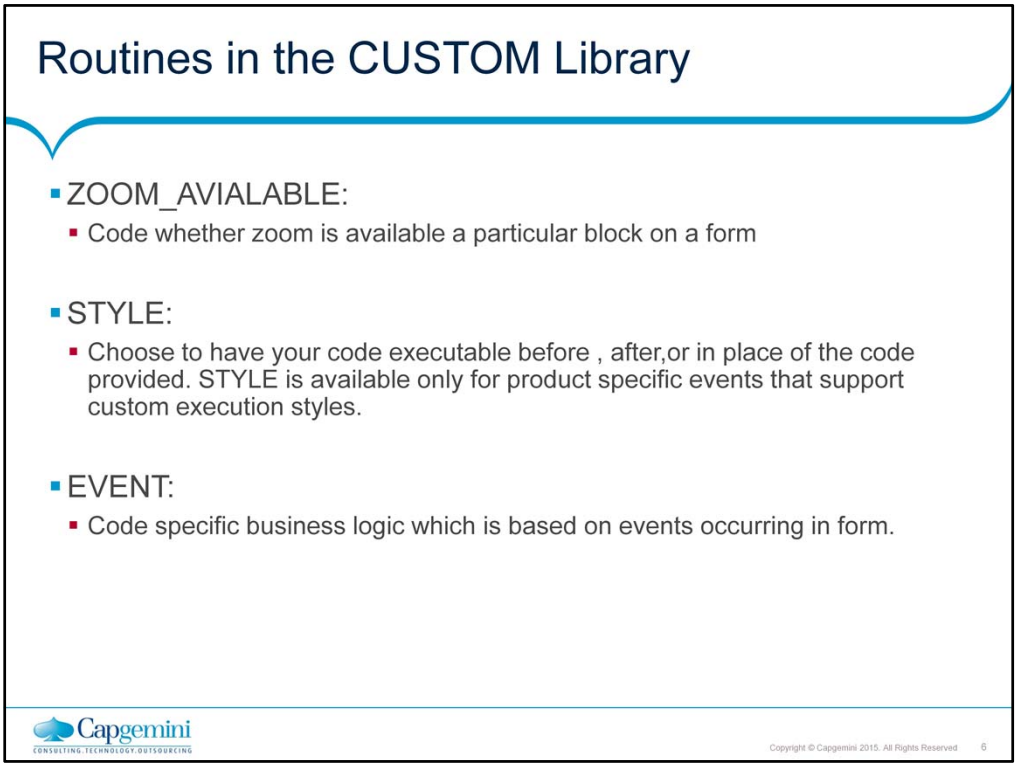
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4

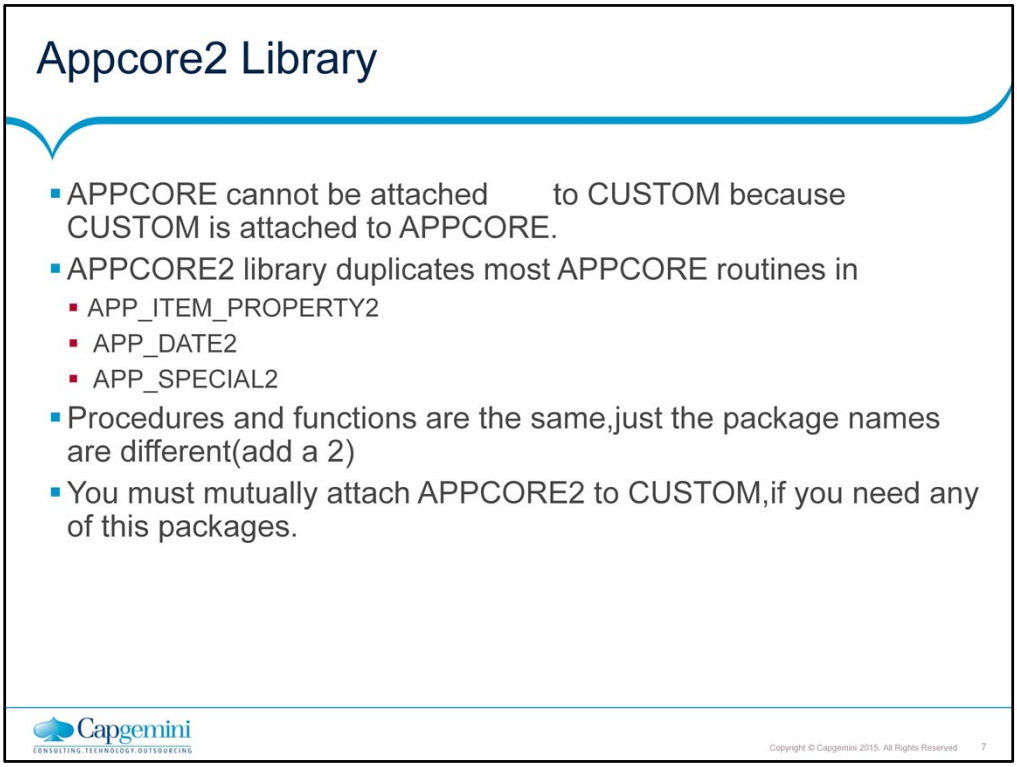
All the forms that are based on the oracle applications TEMPLATE from reference objects from the APPSTAND.fmb form .libraries attached directly to the form include APPCORE,APPYPK,FNDSQF and any other form specific libraries.
Since the CUSTOM library is a typical Forms PL/SQL library, it can include standard PL/SQL logic, Oracle Forms
built-in commands and may have other Forms PL/SQL libraries attached to it. The base CUSTOM library is located
in the $AU_TOP/resource directory on your forms server. Explore the CUSTOM.pll using the Oracle Forms
Designer module to examine the sample code that exists in library. Once any enhancements are made, the library
must be generated into an executable library (CUSTOM.plx) that then can be used by the Oracle Applications Forms
runtime module. Since the CUSTOM library is loaded once for a given session, a user must log out of the
application and sign-on again before any changes will become apparent. It is suggested that you also place a copy of
the CUSTOM library in your customization directory as a safety precaution.
The CUSTOM PL/SQL library does have its limitations. It can not contain SQL commands. It can not have certain
Oracle Applications Forms global libraries attached to it (such as APPCORE). The CUSTOM library is designed to
be used solely with the Oracle eBusiness Applications and does not support the Oracle Self-Service Web
Applications (OSSWA). Since the CUSTOM library's standard location is in the $AU_TOP/resource, it can be
overwritten during an upgrade of the Applications.



The CUSTOM library allows extension of Oracle Applications without modification of Oracle Applications code. You can use the CUSTOM library for customizations such as Zoom (such as moving to another form and querying up specific records), enforcing business rules (for example, vendor name must be in uppercase letters), and disabling fields that do not apply for your site.
You write code in the CUSTOM library, within the procedure shells that are provided. All logic must branch based on the form and block for which you want it to run. Oracle Applications sends events to the CUSTOM library. Your custom code can take effect based on these events.



ZOOM_AVIALABLE
This function allows you to specify if zooms exist for the current context. If zooms are available for this block, then return TRUE else return FALSE. This routine is called on a per-block basis within every Applications form from the WHEN-NEW-BLOCK-INSTANCE trigger. Therefore, any code that will enable Zoom must test the current form and block from which the call is being made. By default this routine must return FALSE.

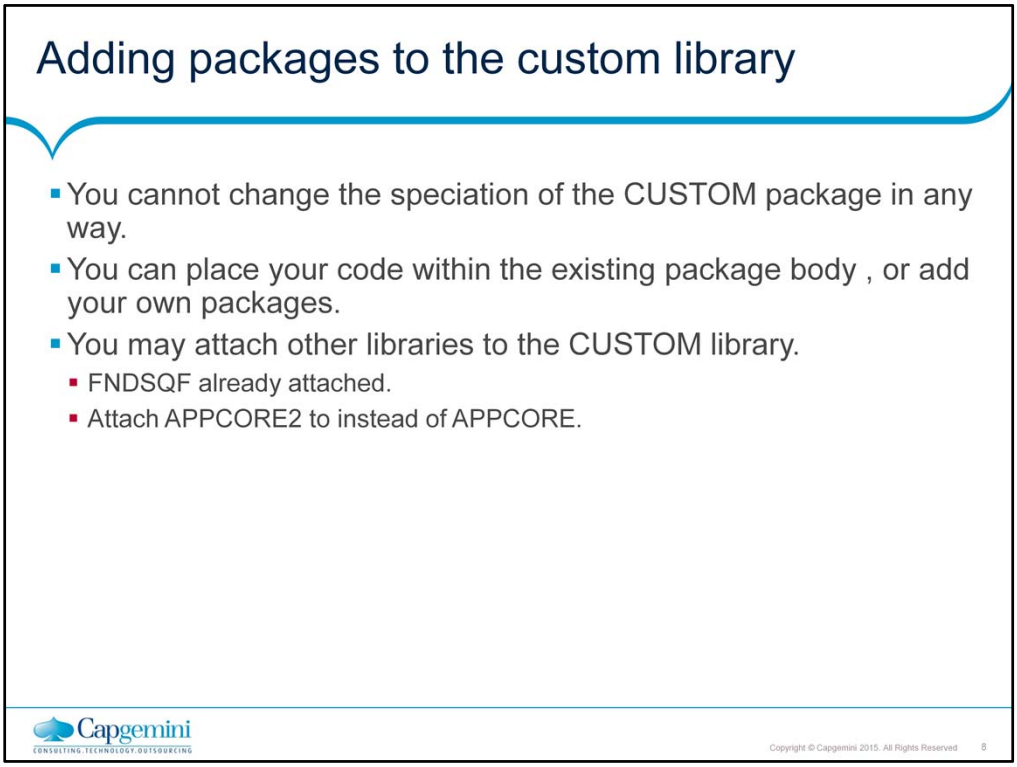


STYLE
This function returns a integer value. This function allows to override the execution style of Product specific events, but it doesn't effect generic events like when-new-form-instance. Possible return values are:
1. custom.before
2. custom.after
3. custom.override
4. custom.standard
By default it returns custom.standard.

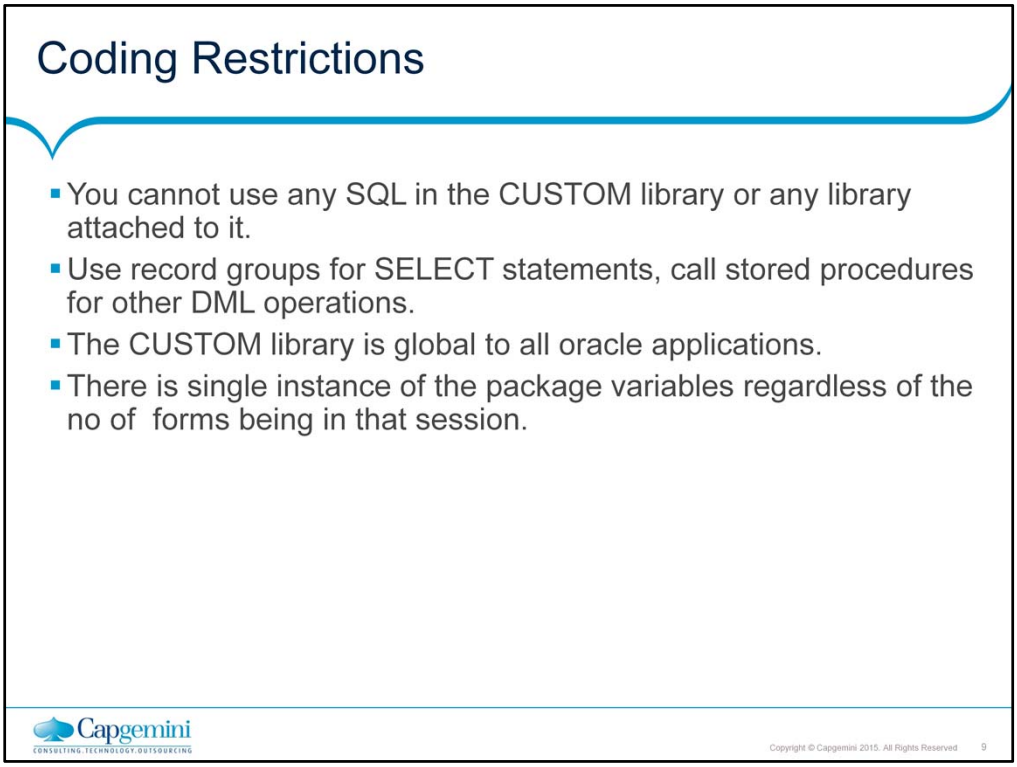
EVENT
This procedure allows you to execute your code at specific events including:

- ZOOM
- WHEN-NEW-FORM-INSTANCE
- WHEN-NEW-BLOCK-INSTANCE
- WHEN-NEW-RECORD-INSTANCE
- WHEN-NEW-ITEM-INSTANCE
- WHEN-VALIDATE-RECORD

By default this routine must perform 'null;'



The CUSTOM library is shipped with the FNDSQF library already attached however, early versions of release 10SC did not include the attachment.



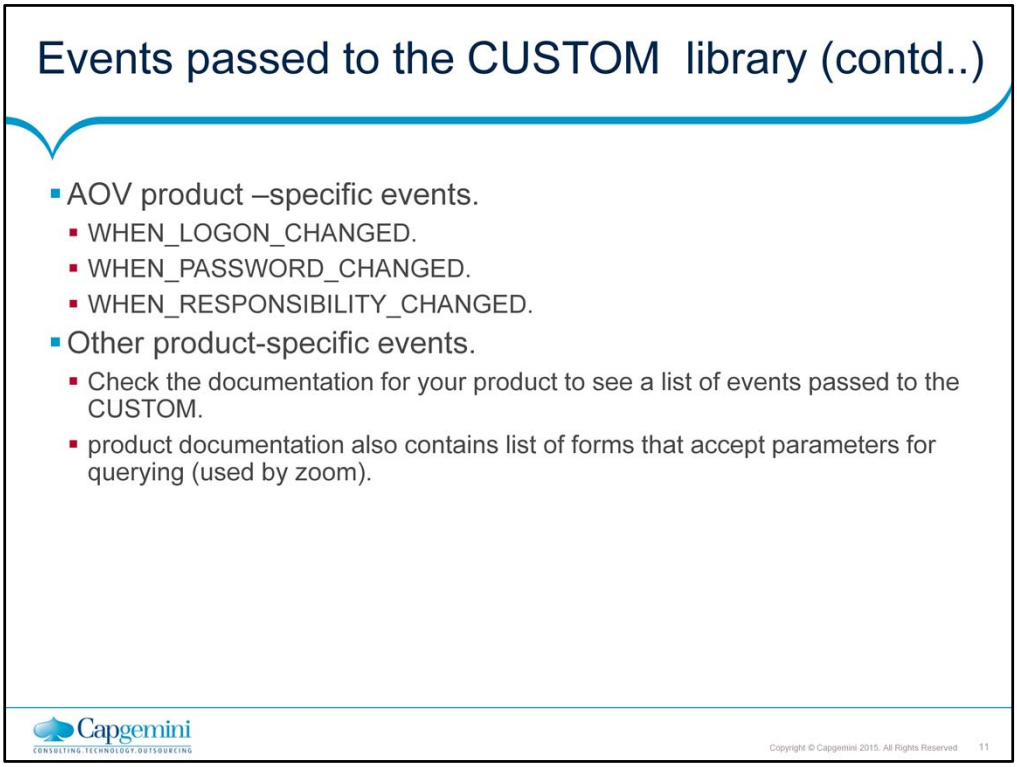
Since the CUSTOM library cannot contain SQL commands, that type of logic must be passed off to stored procedures or functions within the Oracle database.
Within the CUSTOM library, you are free to write almost any code supported by the Developer/2000 toolset, so long as you follow all Oracle Applications coding standards. However, there is an exception: you cannot call any APPCORE routines from CUSTOM. Almost all APPCORE routines start with the prefix "APP".
If you use Zoom or the CUSTOM library to invoke forms that you have developed, those forms must adhere completely to all of the Oracle Applications coding standards.
Attention: To invoke another form, use the function security routines. Do not use CALL_FORM since the Oracle Applications libraries do not support it.
After you write code in the CUSTOM procedures, compile and generate the library using Oracle Forms. Then place this library into $AU_TOP/res/plsql directory (or platform equivalent). Subsequent invocations of Oracle Applications will then run this new code.

Events passed to the CUSTOM library

- Zoom.
- WHEN_NEW_FORM_INSTANCE.
- WHEN_FORM_NEVIGAT.
- WHEN_NEW_BLOCK_INSTANCE.
- WHEN_NEW_RECORD_INSTANCE.
- WHEN_VALIDATE_RECORD.
- WHEN_NEW_ITEM_INSTANCE.
- SPECIAL n and KEY Fn.
- EXPORT.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10



WHEN-LOGIN-CHANGED – when a user logs on as a different user
WHEN-RESPONSIBILITY-CHANGED – when a user changes responsibilities
WHEN-PASSWORD-CHANGED – when a user changes their password

Example: Force a field to uppercase

```
begin
If(event_name='WHEN_NEW_FORM_INSTANCE')
then
If (form_name='APXVENDR') then
app_item_property2.set_property('VENODR.NAME'.
CASE_RESTRICTION, UPPERCASE);
end if;
end if;
end if:
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12

Example: Change a button label

```
Begin
If(event_name='WHEN_NEW_FORM_INSTANCE')
then
If (form_name='APIVENDR') then
app_item)property2.set_property('vendr.details',LABEL,'
More Details');
end if;
end if;
end if:
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

Example: Hide a Field

```
begin
If(event_name='WHEN_NEW_FORM_INSTANCE')
then
If (form_name='APIVENDR') then
app_item)property2.set_property('venodr.note',DISPLAY
ED, PROPERTY_OFF);
end if;
end if;
end if:
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

Example: Change a Field prompt

```
begin
If(event_name='WHEN_NEW_FORM_INSTANCE')
then
If (form_name='APXVENDR') then
app_item_property2.set_property('vendor.name'.PROMPT
_TEXT, 'supplier name');
end if;
end if;
end if:
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

Example: Prevent inserts/updates to a block

```
begin
If(event_name='WHEN_NEW_FORM_INSTANCE')
then
If (form_name='APIVENDR' and block_name='VENDOR') then
Set_block_property(block_name,insert_allowed,property_false)
;
Set_block_property(block_name,update_allowed,property_fals
e);
End if;
End if;
End if;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

Special Menu Example

- (display the special15 entry on the tools menu)

```
begin
If(event_name='WHEN_NEW_FORM_INSTANCE')
If(form_name='DEMXXOER') then
App_special2.instantiate('SPECILA15','print order
again','',TRUE,'LINE');
End if;
End if;
End if;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17

Special Menu Example

- (toggle display the special15 tools menu entry.)

```
If(event_name='WHEN_NEW_FORM_INSTANCE')
If(form_name='DEMXXOER' and
block_name='ORDERS')then) then
App_special2.enable('SPECILA15',PROPERTY_ON);
Elseif(form_name= 'DEMXXOER' and
block_name='LINES')then
App_special2.enable('SPECILA15',PROPERTY_OFF);
End if;
End if;
End if;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 18

Special Menu Example

- (code the logic for the special15 menu entry)

```
If(event_name='SPECIALS')then
If(form_name='DEMXXEOR' and block_name='ORDERS') then
/*add your print order logic here*/
Raise FORM_TRIGGER_FAILURE;
End if;
End if;
End if;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19

Coding CUSTOM.ZOOM_Available

- (If zoom is available for this block then TRUE; otherwise return FALSE.)

```
Function zoom_available return boolean is form_name
varchar2(30):=name_in( 'system.current_form');
Block_name varchar2(30):=name_in(
'system.cursor_block');
Begin
If(form_name='FNDSCAUS' and block_name='USER') then
Return TRUE;
else return FALSE;
End if;end zoom_avialable;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20

CUSTOM.EVENT for ZOOM

```
begin
If (event_name= 'ZOOM')then
If (form_name='FNDSCAUS' and block_name='USER') then
Param_to_pass:=name_in('user.description');
Fnd_function.execute(FUNCTION_NAME=>'DEM
DEMXXOER',OPEN_FLAG=>'Y',SESSION_FLAG=>'Y',OTH
ER_PARAMS=>'DESCRIPTION="'||param_to_pass||' " ');
End if;else
Null;end if;end;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

Support

- There is a new LOV and parameter in all 11i forms:
- APPCORE_ZOOM(LOV)
- APPCORE_ZOOM_VALUE (parametre)
- Use these when you have more than one zoom from a perticular block.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

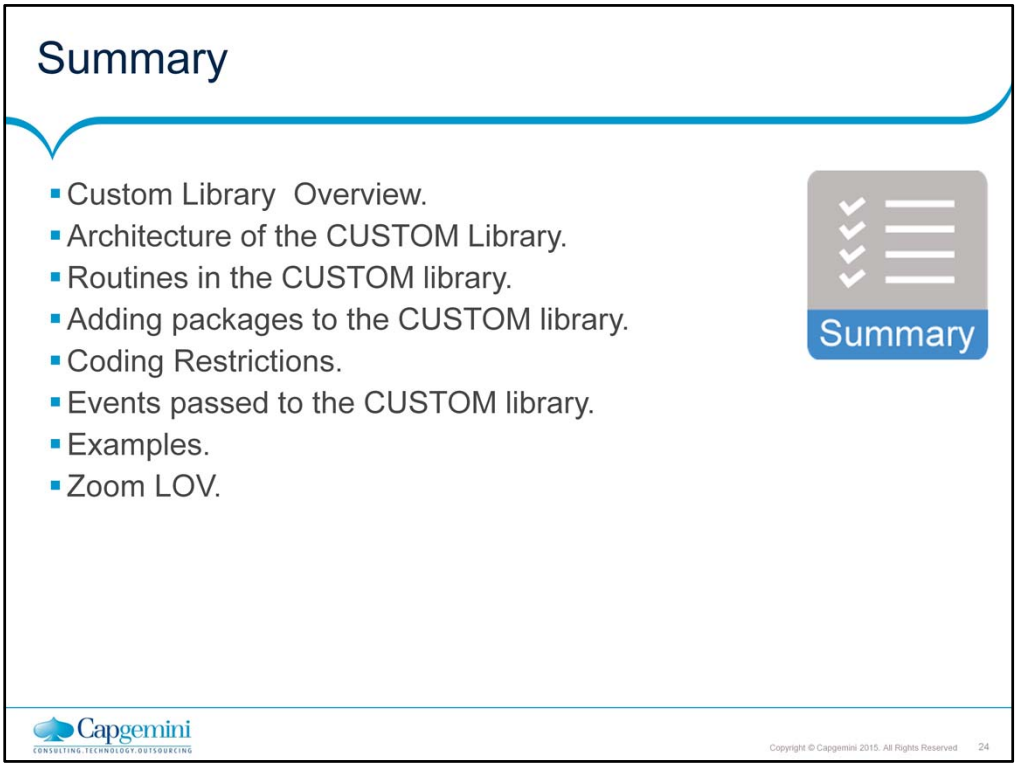
 22

Zoom LOV

- Create a record group in CUSTOM library and populate it with the names of available zooms for the block.
- Attach the record group to the APPCORE_ZOOM LOV.
- Call show_lov to show the APPCORE_ZOOM LOV.
- When the user picks a zoom, a value is returned into the APPCORE_ZOOM_VALUE parameter in the form.
- Retrieve the value and branch your zoom code accordingly.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23



Add the notes here.

Review Question

- Question 1: CUSTOM contains how many routines?
  - Option 1: 1
  - Option 2: 2
  - Option 3: 3
- Question 2: APPCORE can be attached to CUSTOM library
  - True/False
- Question 3: Where CUSTOM.pll is located?
  - Option 1: $AU_TOP/resource Directory.
  - Option 2: Fnd_top.
  - Option 3: Custom_top.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 25

Add the notes here.

ERP - Oracle Apps

Lesson 19: Personalization of Forms

Lesson Objectives

- Overview
- Terminology
- Personalizing a Form
- Defining Rules
  - Rules
  - Conditions
  - Scope
  - Actions
- Evaluation of Strings
- Relationship to Custom Library
- Limitations

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

Overview

- The Form Personalization feature is introduced in 11.5.10
- Ths allows you to declaratively alter the behavior of Forms-based screens, including changing properties, executing builtins, displaying messages, and adding menu entries.
- For each function, you can specify one or more Rules.
- Each Rule consists of an Event, an optional Condition, the Scope for which it applies, and one or more Actions to perform.
- Once Rules are defined, when the target function is run then the Rules are automatically applied as events occur within that form.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

Terminology

- Event
  - An Event is a trigger point within a form, such as WHEN-NEW-FORM-INSTANCE, WHEN-NEW-RECORD-INSTANCE.
  - There are standard events that almost every form sends, and certain forms send additional product-specific events.
- Scope
  - The Scope is evaluated based on the current runtime context to determine if a Rule should be processed or not.
  - The Scope can be at the Site, Responsibility, User, or Industry level. Each Rule can have one or more Scopes associated with it.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4

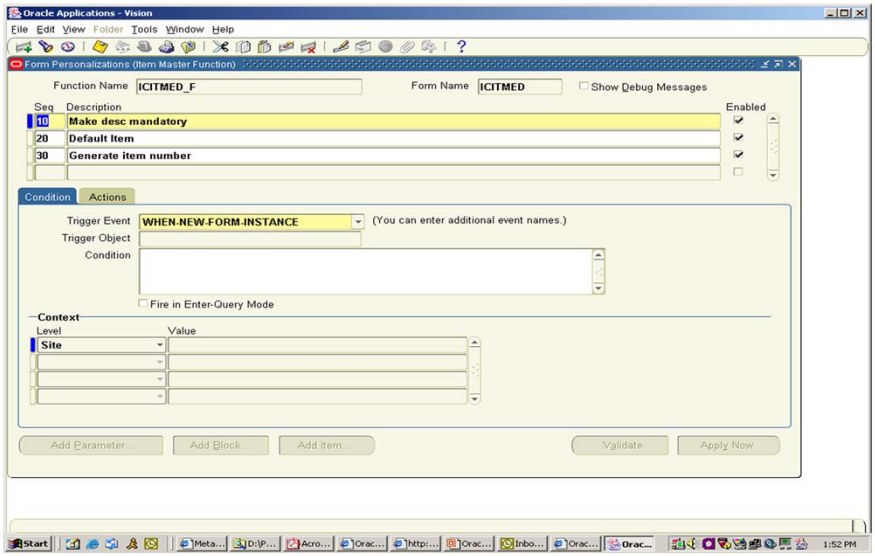
Terminology

- Condition
  - The Condition is an optional SQL code fragment that is evaluated when the Event occurs; if it evaluates to TRUE then the Actions are processed.
- Action
  - Each Action consists of one of the following:
    - setting a Property, such as making a field Required or hiding a Tab page
    - executing a Builtin, such as GO_BLOCK, DO_KEY or FND_FUNCTION.EXECUTE
    - displaying a Message
    - enabling a Special menu entry

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

Personalizations Form



Personalizing a Form

- To create personalizations for a particular function, first invoke that function from the Navigation menu.
- While in the form, choose Help->Diagnostics->Custom Code-> Personalize from the pulldown menu.
- The Personalization form will open and automatically query existing Rules for that function.
- After making changes, Save them then close and re-run the function to have them take effect.
- You can also Validate or Apply certain changes immediately to test them without having to re-run the target form by pressing the 'Validate' or 'Apply Now' buttons.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

Personalizing a Form

- This menu entry is secured by the 'Hide Diagnostics' menu entry and 'Utilities:Diagnostics' profiles, as are most other entries on the Diagnostics menu.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

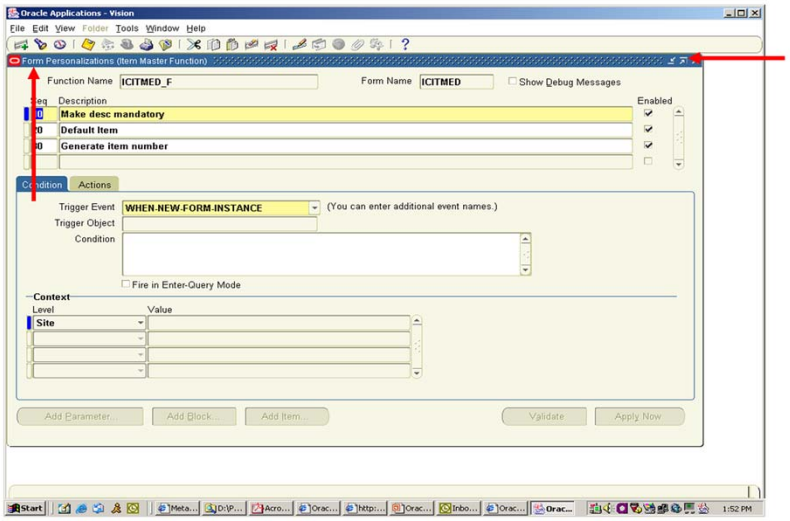
Defining Rules

- Each Rule consists of the following fields:
  - Seq: The sequence in which rules will be processed. This is a value between 1 and 100, with 1 being processed first. The sequence of rules does not have to be unique. Note that there is an interaction with the Trigger Event field, described below.
  - Description: Use this field to document the personalization you are making.
  - Enabled: Uncheck this checkbox to temporarily disable processing of a Rule.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

Defining Rules



Defining Rules - Conditions

- Condition tab:
  - Trigger Event: Select the event at which you want the Rule to be processed.

    Rules are processed first by matching the Event, then by their Sequence number.
  - Trigger Object: Depending on the Trigger Event, this field may be Disabled, or Enabled and Required in which case it will validate against a List of Values.

    For example, if Trigger Event WHEN-NEW-ITEM-INSTANCE is selected, then you must enter a specific block.field for that trigger to be processed.
  - Condition: This is an optional SQL code fragment that is evaluated when the Event occurs; if it evaluates to TRUE then the Actions are processed.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

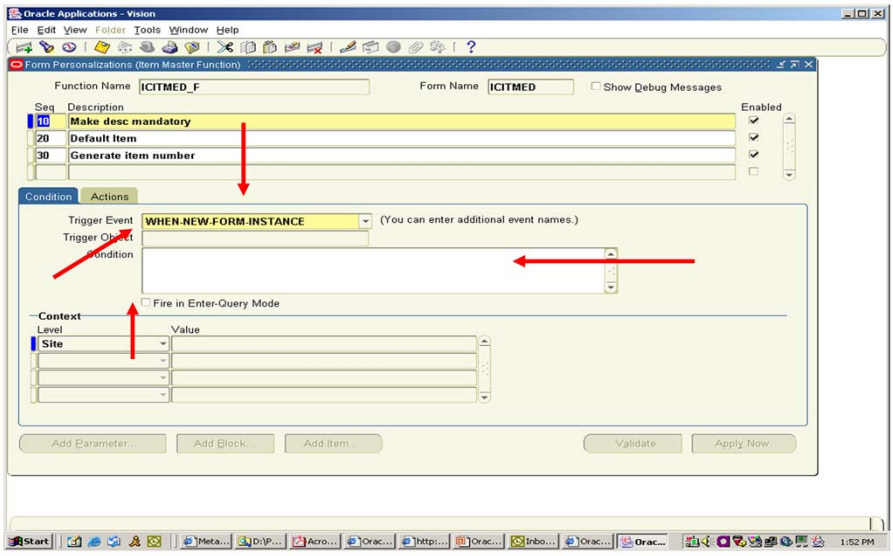
Defining Rules - Conditions

- The condition can contain any of the following:
  - SQL functions and operators, such as AND, OR, TO_CHAR, DECODE, and NVL
  - References to bind variables (:block.field), including :system, :global and :parameter values.
    - Use the 'Add Item...' button to assist with item names.
  - Calls to server-side functions that do not have OUT parameters
  - The syntax can be tested with the 'Validate' button, which will evaluate it in the current context of the target form.
  - Fire in Enter-Query Mode: This checkbox controls whether the Rule should be processed if the event occurs during enter-query mode processing.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12

Defining Rules - Conditions



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

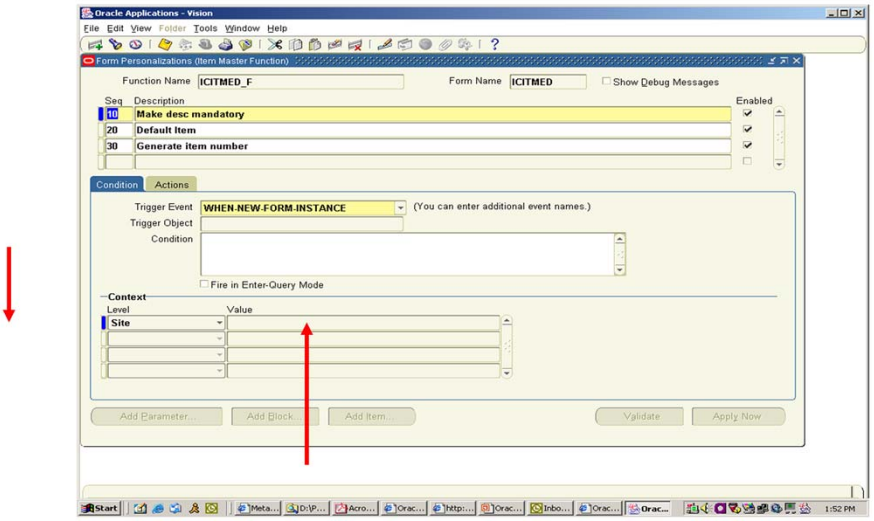
Defining Rules - Scope

- The following Scope fields appear in the Context region of the Condition tab:
  - Level: Select the level at which you want the rule to be applied, either Site, Responsibility, User, or Industry.
  - Value: Based on the Level, either Disabled, or Enabled and Required in which case it will validate against a List of Values.
  - If a Rule has no Scope rows or Action rows, it is not processed.
  - Upon saving a Rule, if no Scope rows have been entered the form will automatically create a row at the Site level.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

Defining Rules - Scope



Defining Rules - Actions

- Actions tab:
  - Seq: The sequence in which actions will be processed within that Rule. This is a value between 1 and 100, with 1 being processed first. The sequence does not have to be unique.
  - Type: The type of action to take
    - Property: Allows you to select a specific object, a property of that object, and specify a new value for that property
    - Builtin: Allows execution of a standard Forms Builtin, such as GO_BLOCK or DO_KEY
    - Message: Displays a message in one of several styles
    - Special: Enables a special menu entry, defining its label, icon name and which blocks it applies to.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

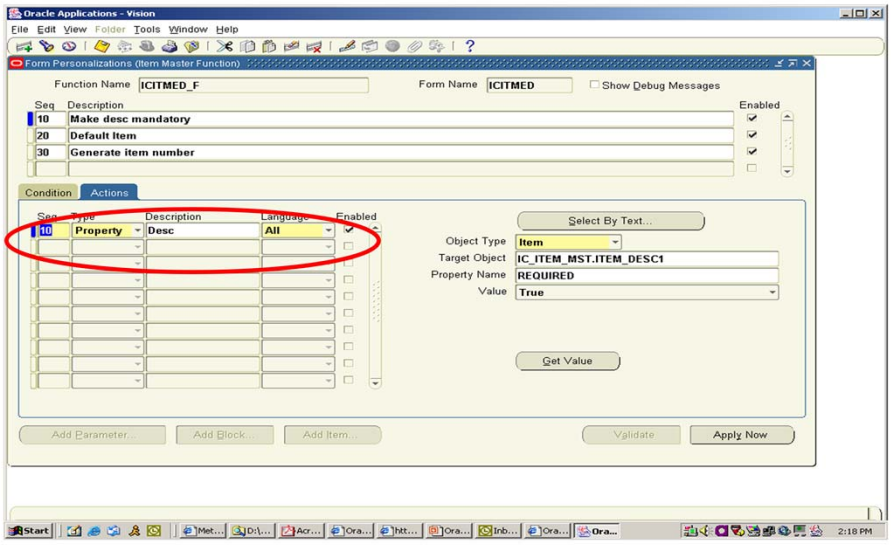
Defining Rules - Actions

- Description: Use this field to document the personalization action you are making.
- Language: Specify 'All' to have the action processed for any language, or select a specific language. Typically text-related personalizations would be applied for a specific language.
- Enabled: Uncheck this checkbox to temporarily disable processing of the action.
- Apply Now: For several Types, this button will be enabled. It allows you to apply the change immediately to the target form to test its effect.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17

Defining Rules - Actions



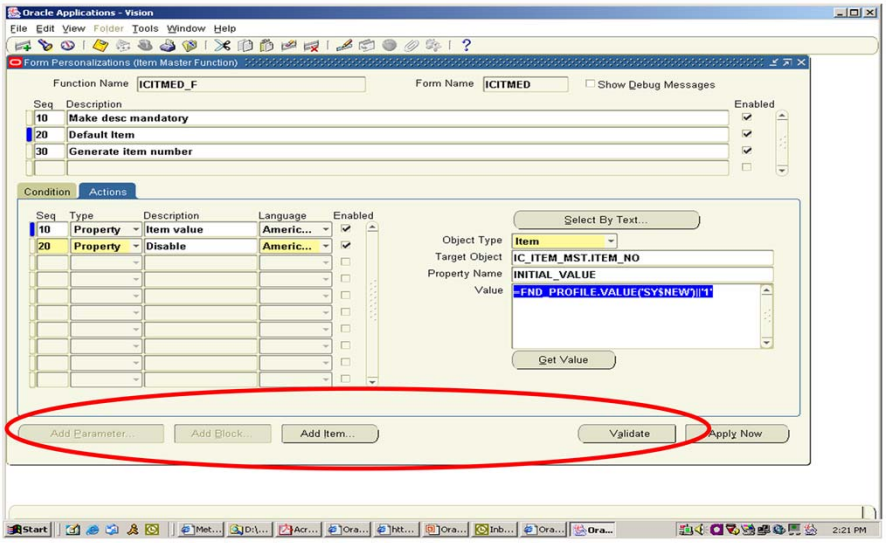
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 18

Define Rules - Actions

- The following buttons are enabled conditionally based on the Type field:
  - Add Parameter: List of Values that displays currently used parameters. Applies to the builtin FND_FUNCTION.EXECUTE only.
  - Add Block: List of Values that displays block names.
  - Add Item: List of Values that displays item names.
  - Validate: Used to test if the syntax of your string is valid.
- If the evaluation fails, the processing engine will return an ORA error.Otherwise, it will display the text exactly as it would appear at runtime in the current context.

Define Rules - Actions



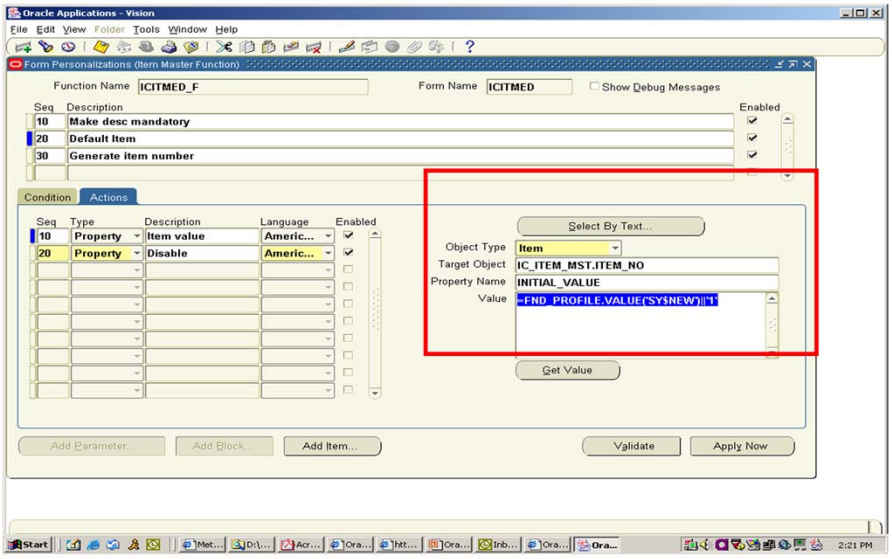
Defining Rules - Actions

- Some fields appear conditionally based on the Type field.
  - Type: Property
    - Select By Text: Allows you to select an object based on text appearing on the screen at the point in time that you invoke the Personalization form, including any changes that current rules might have performed.
    - Selecting it will automatically fill in the Object Type and Target Object fields.
    - Object Type: The type of object, including Item, Window, Block, Tab, Canvas, Radio button, View, GLOBAL, or PARAMETER.
    - Target Object: The internal name of the object. For Object Types of GLOBAL and PARAMETER, the Target Object name must not include those keywords.
    - Property Name: The properties that can be personalized.
    - Value: The new value.
    - Get Value: This button gets the current property value of the object.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

Defining Rules - Actions



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22

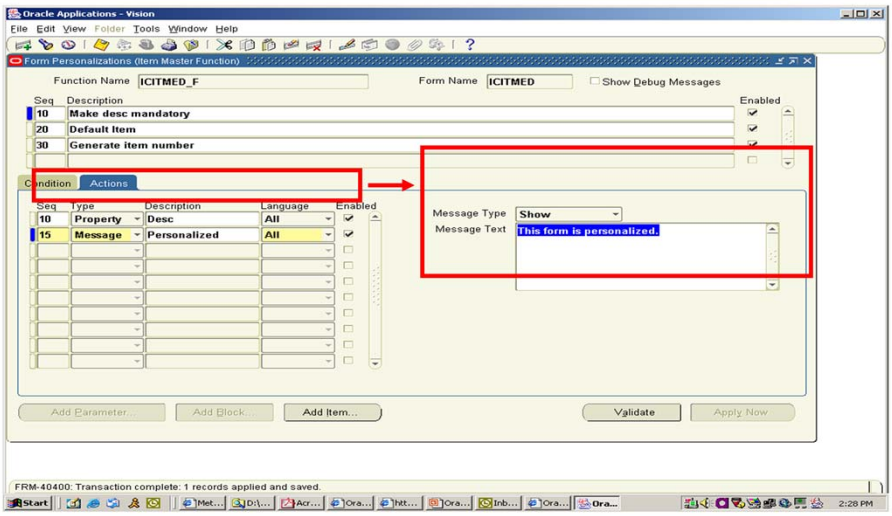
Defining Rules - Actions

- Type: Message
  - Message Type: Either 'Show', 'Hint', ‘Warn’, 'Error', or ‘Debug’.
  - ‘Error’ and ‘Warn’ if the user selects the ‘Cancel’ button will raise a form_trigger_failure after executing, and stop all further processing.
  - Messages of type ‘Debug’ will only be displayed if the ‘Show Debug Messages’ checkbox is checked.
  - Message Text: The text you want to display in the message.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

Defining Rules - Actions



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 24

Defining Rules - Actions

- Type: Built-in
  - Builtin Type: The name of the builtin, such as
  - GO_ITEM
  - DO_KEY
  - GO_BLOCK
  - RAISE FORM_TRIGGER_FAILURE
  - FORMS_DDL
  - FND_UTILITIES.OPEN_URL
  - FND_FUNCTION.EXECUTE
  - Argument: The argument for the currently selected builtin, if applicable.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 25

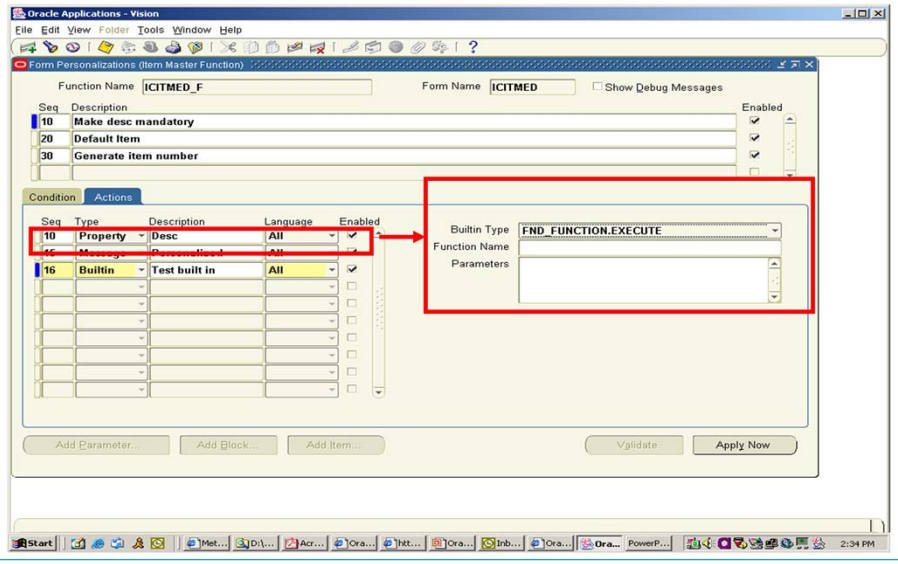
Defining Rules - Actions

- Depending on the specific builtin, other argument fields may appear.
  - Function Name: The name of the function that should be executed.
  - Parameters: You can manually enter parameters or use the 'Add Parameter…' button.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 26

Defining Rules - Actions



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 27

Defining Rules - Actions

- Type: Special
  - Menu Entry: One of 45 menu entries that you can activate.
  - Menu Label: The textual label that you want on the menu entry.
  - Render line before menu: If checked, will render a line above the menu entry to visually separate it from prior entries.
  - Enabled in Block(s): Specify the blocks that you want the menu entry enabled in; specify more than one block by separating the names with a comma.
  - If no blocks are specified, the entry will be enabled in all blocks.
  - Icon Name: Specify an optional icon name that you want added to the Toolbar to achieve the same functionality as in the special pulldown menu entry.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 28

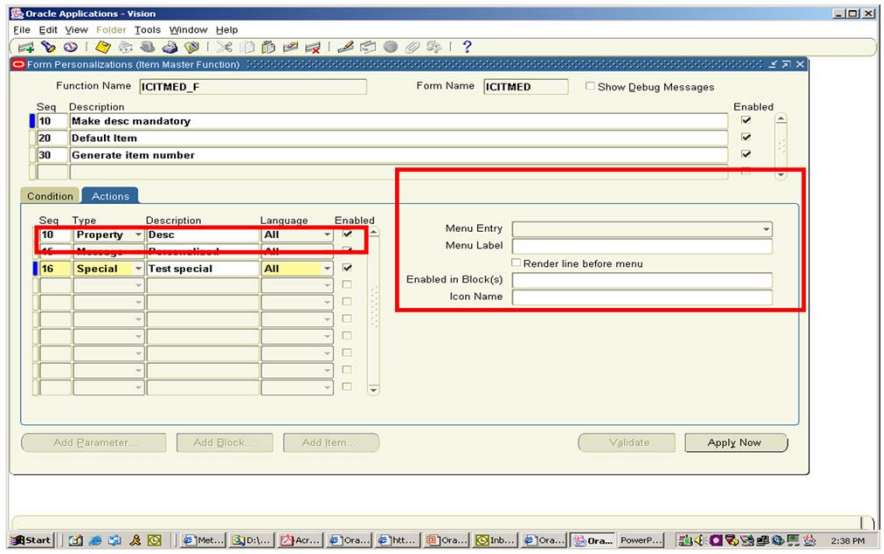
Defining Rules - Actions

- Specifying an action of 'Special' merely activates the appropriate menu entry.
- When the user selects the entry, it will fire the corresponding SPECIAL# trigger.
- You must must also create another rule that traps this Trigger Event and performs the desired functionality.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 29

Defining Rules - Actions



Evaluation of Strings

- Every property that takes a string can either be processed literally or evaluated at runtime.
- If you type a string in that does not start with '=', then the exact value you type in will be used at runtime.
- If the string you type starts with '=', then the text immediately after that character will be evaluated at runtime.
- This allows you to write complex logic that can include references such as:
  - SQL operators, such as ||, TO_CHAR, DECODE, and NVL
  - bind variables (:block.field), including :system, :global and :parameter values.
  - Calls to server-side functions that do not have OUT parameters.
  - SELECT statements.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 31

Evaluation of Strings

- To use the SELECT form, you must follow these rules:
  - The text must start with '=SELECT'
  - The column being selected must evaluate to a CHAR, with a length no longer than 2000 bytes.
  - You must alias the column being selected to 'A'.
  - Your SELECT statement should only return one row, but if more than one is returned only the value of the first row will be used.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 32

Relationship to CUSTOM Library

- Form Personalization allows personalizations that could be made in the CUSTOM library.
- It does not require that you use the Oracle Forms Builder to edit and compile the CUSTOM file.
- The CUSTOM library is able to support more complex personalizations because it gives you access to:
  - all of the capabilities of the PL/SQL programming language
  - client-side program units
  - all Oracle Forms builtins
  - issuing any SQL.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 33

Relationship to CUSTOM Library

- Both Form Personalization and the CUSTOM library drive off the exact same events.
- The Form Personalization feature receives and processes them first, then passes them to the CUSTOM library, thus you can use both mechanisms simultaneously.
- Both features also respond identically to the Custom Code events of 'Normal', 'Off' and 'Core Code Only'.
- In general, Oracle recommends that you use the Form Personalization feature whenever possible, and only use the CUSTOM library when significantly more complex processing is required.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 34

Limitations

- Form Personalization can only respond to events that are centrally processed and dispatched by APPCORE. These are limited to:
  - WHEN-NEW-FORM-INSTANCE
  - WHEN-NEW-BLOCK-INSTANCE
  - WHENNEW-RECORD-INSTANCE
  - WHEN-NEW-ITEM-INSTANCE.
  - WHEN-VALIDATE-RECORD
  - SPECIAL1 through SPECIAL45.
  - Product-specific events. These are typically documented in implementation manuals.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 35

Limitations

- You can only change what Oracle Forms allows at runtime. For example, the following cannot be changed:
  - You cannot create new items
  - You cannot move items between canvases
  - You cannot display an item which is not on a canvas (thus, individual flexfield segments cannot be displayed)
  - You cannot set certain properties such as the Datatype of an Item.
  - You cannot change frames, graphics, or boilerplate
  - You cannot hide the item that currently has focus

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 36

Demo

- Disable the Submit button of order entry form using form personalization.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 37

Summary

- In this lesson, you should have learned how to do the following:
  - Overview
  - Terminology
  - Personalizing a Form
  - Defining Rules
  - Evaluation of Strings
  - Relationship to Custom Library
  - Limitations

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 38

Review Questions

- Question 1: Which of the following activity cannot be done using Form Personalization :
  - Create new items
  - Move items between canvases
  - Disable Items
  - Hide the item that currently has focus

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 39

ERP- Oracle Apps

Lesson 20: Workflow

Lesson Objectives

- To understand the following topics:
- Workflow
  - Overview of Oracle Workflow
- Components
  - Workflow Engine
  - Workflow Monitor
  - Workflow Builder
  - Notification Mailer
  - Directory Services
- Workflow Examples (Using Workflow Builder)
  - Creation Of new Workflow
  - Approaches for tracking Errors
- Workflow Related screens/Programs in APPS

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

What is Oracle Workflow

- Graphically represents a business process using drag and drop process designer.
- Oracle Workflow comprises of a set of tables , PL/SQL packages, Procedures ,front-end tools and back-end processes that are initiated automatically in the background as and when required according to the business rules modeled.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

Why Workflow ?

- Flexibility in modeling, automation, and continuous improvement of business processes, routing information of any type according to user–defined business rules.

- Benefits from Oracle Workflow
  - Routing Information
  - Defining and Modifying Business Rules
  - Delivering E-Notifications
  - Integrating Systems by using Business Event Manager.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

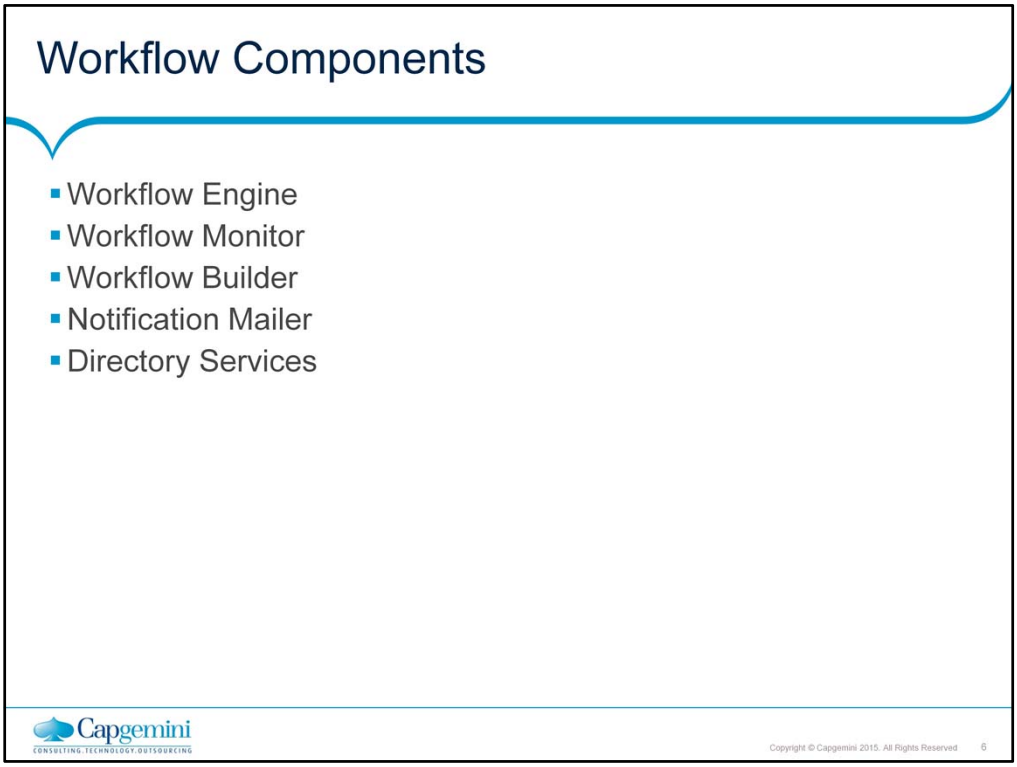
 4

Oracle Workflow Overview

- Oracle Workflow accomplishes three important business requirements :
  - Routes information
  - Defines any business rule
  - Delivers electronic notification

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5


Workflow Components
Workflow Engine
Workflow Monitor
Workflow Builder
Notification Mailer
Directory Services
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
6

Oracle Workflow Engine

- The Workflow Engine embedded in the Oracle database server, implements process definitions at runtime.
- The Workflow Engine monitors workflow states and coordinates the routing of activities for the process.
- Changes in workflow state, such as the completion of Workflow activities, are signaled to the engine via a PL/SQL API or a Java API.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

Oracle Workflow Engine Features

- Based on flexibly–defined workflow rules, the engine determines which activities are eligible to run, and then runs them.
- The Workflow Engine
- supports sophisticated workflow rules, including looping, branching, parallel flows, and sub flows.
- Can defer costly function activities to background engines for processing
- Maintains a history of completed activities
- Detects error conditions and runs error processes

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

Initiating a Workflow Process

- To start a workflow process:
  - Your application must execute a procedure that calls the following Workflow Engine API's
  - WF_ENGINE.CreateProcess
  - WF_ENGINE.StartProcess
  - The procedure must identify the value of the process item type and item key for these API's
  - These API's can be called from Forms,PLL's back-end procedures ,functions,triggers and packages

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

Workflow Engine Processing

- Upon starting a process , the workflow engine:
  - Identifies and executes the Start activity
  - Determines the next activity to transition to after completing the prerequisite activity or activities
  - Drives through the process, automatically executing all function activities, until it comes to a notification activity or blocking activity
  - Calls the notification system to notify the performer
  - Continues driving through the process until it encounters an End activity.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

Example: Initiating a Workflow Process

```
procedure StartProcess (RequisitionNumber in varchar2,
                        RequisitionDesc in varchar2,
                        RequisitionAmount in number,
                        RequestorUsername in varchar2,
                        ProcessOwner in varchar2,
                        Workflowprocess in varchar2 default null,
                        item_type in varchar2 default null) is

  ItemType    varchar2(30) := nvl(item_type,'WFDEMO');
  ItemKey             varchar2(30) := RequisitionNumber;
  ItemUserKey         varchar2(80) := RequisitionDesc;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

Example: Initiating a Workflow Process (Contd…)

```
begin
                wf_engine.CreateProcess     (...);
                wf_engine.SetItemUserKey    (...);
                wf_engine.SetItemAttrText   (...);
                wf_engine.StartProcess      (...);
exception
        when others then
                wf_core.context (...);
                raise;
end StartProcess;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12

Background Engine

- The Background Engine is a PL/SQL procedure which executes the deferred activities.
  - The Background Engine checks for and executes any deferred or timed out activities that satisfy the arguments of the procedure at the time the procedure is invoked.
  - The procedure ends once all matching activities are executed.
  - If new activities are deferred or timed out after the initiation of the current Background Engine procedure, they will be processed when the next Background Engine procedure is initiated.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

Background Engine (Contd…)

- Background Engine API

```
WF_ENGINE.BACKGROUND(
itemtype                in varchar2,
minthreshold            in number default null,
maxthreshold            in number default null,
process_deferred        in boolean default TRUE,
process_timeout         in boolean default TRUE);
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

Deferred Processing

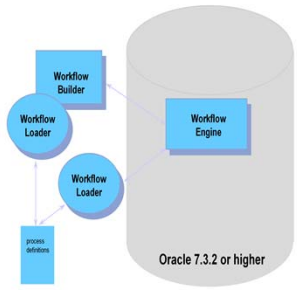
- Set the Workflow Engine threshold cost to control which activities get deferred.
- The default threshold cost of the workflow engine is set to 50.
- The workflow engine threshold is an externalized constant.
- Add the command to a PL/SQL stored procedure or execute the command in sql*plus to change the threshold :
  WF_ENGINE.THRESHOLD := n;

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

Workflow Definitions Loader

- The Workflow Definitions Loader is a utility program that moves workflow definitions between database and corresponding flat file representations.

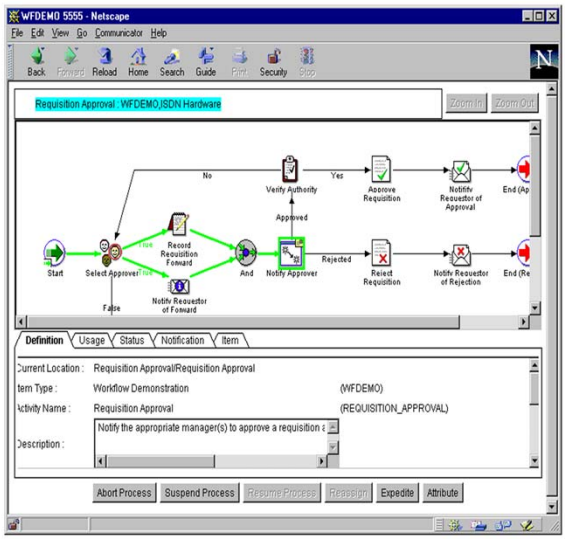


Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

Workflow Monitor

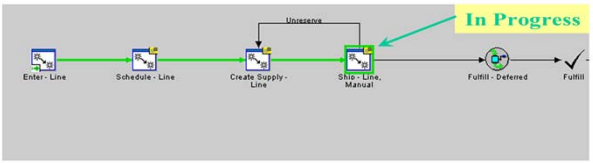
- The Workflow Monitor displays an annotated view of the process diagram for a particular instance of a workflow process, so that users can get a graphical depiction of their work item status.



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17

Workflow Monitor: Process Diagram



Color	Status
Green	In Progress
Red	Error
Yellow	Hold
<None>	Complete

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 18

Workflow: Notifications

- Automatically Notify Users
  - Alert users of exception conditions
- Respond to events that require human judgment
  - Send Notifications to individuals or roles
  - Change the participants in a role without changing the process
- Automatic Notification Forwarding
  - When user is unavailable (vacation, sick leave)
  - When user wants to temporarily delegate a task

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19

Overview of the Workflow Directory Service

- Workflow uses Directory Service to resolve the performers and email addresses for sending the notifications
- Oracle Workflow ships a unified directory service for Oracle Applications-wfdirhrv.sql. The unified directory service maps:
- WF_USERS to Oracle HRMS employees, Oracle Application/Oracle Self-service application users, oracle receivables customer contacts and WF_LOCAL_USERS.
- WF_ROLES to users in WF_USERS , Oracle HRMS positions, oracle applications responsibilities, oracle engineering approval lists and WF_LOCAL_ROLES

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20

Directory Service Views And Local Tables

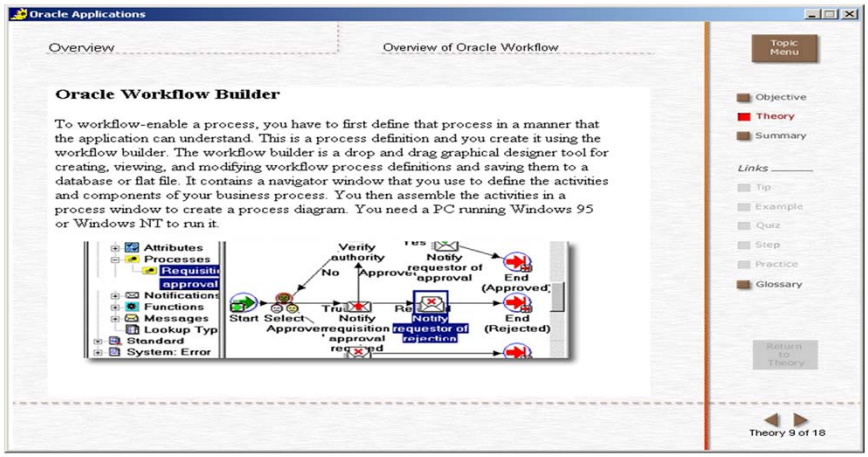
- WF_USERS
- WF_ROLES
- WF_USER_ROLES
- WF_LOCAL_USERS
- WF_LOCAL_ROLES
- WF_LOCAL_USER_ROLES

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

Workflow : Workflow Builder

- Graphical Drag & Drop Designer Tool



Notifications

- A notification activity sends a message to a user or role.
- Oracle Workflow sends a notification to a role when the Workflow Engine executes a notification activity in a workflow process. The notification activity may designate the role as being responsible for performing some human action or may simply relay process–related information to the role.
- Messages lists the messages that a notification activity is associated with. the current item type can send to a user or role. A message can have message attributes associated with it.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

Summary

- Oracle Workflow manages business processes according to rules that you define
- Define your business rules using Oracle Workflow Builder and PL/SQL
- The Workflow Engine and the Notification System carry out your business rules
- Any Oracle Applications user and anyone with access to the internet can participate in the workflow.
- Use the Workflow Monitor to check and/or administer the status of your workflow processes.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 24

Review Questions

- Question 1: Which of the following activity cannot be done using Form Personalization :
  - Create new items
  - Move items between canvases
  - Disable Items
  - Hide the item that currently has focus

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 25

Review Questions

- Question 2: Which of the following is NOT the component of workflow?
  - Workflow Engine
  - Workflow Monitor
  - Workflow Tracker
  - Notification Mailer

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 26