GL Interface

Procedure:

1) We populate the Staging Table
2) Performed Validation on Staging Table.
3) Populate GL Interface
4) Used Concurrent Program : Journal IMPORT.
5) Report Used is : Journal Import Execution Report – That gives Info on failed import.
6) If we get few errors, then we fix the errors in GL_Interface.
7) If we get many Errors: Then we delete IMPORT Journals using
   Concurrent Procss : Delete Journal Import Data.
   - we clean the GL_Interface
   - Fix the data from the Source and start over.
8) Post Journal Entries after successful Import.

Interface Tables: GL_INTERFACE

Base Tables : GL_JE_BACTHES,
GL_JE_HEADERS,
GL_JE_LINES.
[GL_JE_Set_Of_Books, Gl_Code_Combinations, GL_JE_Source_TL, GL_JE_Categoies_TL, GL_Periods, fnd_Currencies]

Staging Table Cols : AccountingDate, DateCreated, ActualFlag, CategoryName, SourceName, Entered Debit/Credit, Segments 1-5, References 21-27,CCID, ProcessFlag, ErrorMsg.
(**N**ew, **Y**es ,**E**rror)

Interface Cols : Status, SetOfBooksID, AccountingDate, CurrencyCode, DateCreated, CreatedBy, ActualFlag, UserJECategorName, UserJESourcename, Entered Debit/Credit, Segments 1-5, References 21-27.

Concurrent Program: Journal Import
Journal Posting --- Populates GL_BALANCES

Validations:-

1) JeHeaderID <GL_JE_Headers>	--	Unique
2) JeBatchID <GL_JE_Batches>	--	Unique
3) JeLineNum <GL_JE_Lines>	--	Uniqueness of HeaderID. -- The Amount of the lines should match the total Amt of Header.
4) SetOfBooksID <GLJESetOfBooks>	--	It has to exist to the set of Books Table.
5) JESourceName <GL_JE_Source_TL>	--	Unique
6)JECategoryName <GL_JE_Categoies_TL>	--	Unique
7) CurrencyCode <Fnd_Currencies>	--	It has to be defined
8) PeriodSetName + Period_Name <GLPeriod>	--	Should be open and defined.
9) CodeCombinationId < Gl_Code_Combinations >	--	Should exist in Chart of Accounts.

Validations Type:

1) Batch Level: a) SOB b) BatchName c) PeriodName
   This is done to ensure that batch doesn't exist already.
2) Journal Level: Journal Entry Name, Currency Code, Accouting Date
3) Accounting Validations / Journal Entry Line level Validation
   Code Combination ID: 1) Should be enabled in Accounting Date.

Validations for the staging table:
Check if the inputted data file is already uploaded into staging table.

Check if the record already exists in the interface table.
Check if the journal already exists in the GL application.

AP Invoice Interface

Interface Table: AP_INVOICES_INTERFACE, AP_INVOICE_LINES_INTERFACE, AP_INTERFACE_REJECTION, AP_INTERFACE_CONTROL

(**AP_INTERFACE_CONTROL**: This is a Temp table that holds control information about segregated data in AP_INVOICES_INTERFACE table during payable open interface import. This table ensure each import is unique With respect of combination of SOURCE & Group_ID.)

Base Table:		
	AP_INVOICES_ALL	-- Header information,
	AP_INVOICE_DISTRIBUTIONS_ALL	-- Lines info
	AP_PAYMENTS_SCHEDULE_ALL	

Staging Table Col: Vendor#, Invoice#, InvoiceAmt, InvoiceDate, Qty, UnitPrice, InvoiceLineAmt, InvoiceLineDesc, Batch#, Status, CreationDate.

Interface Cols:
<**AP_INVOICES_INTERFACE**>: InvoiceID, InvoiceDate, CreationDate, CreatedBy, LastUpdateDate, LastUpdatedBy, InvAm, VendorID, VendorSiteID, Inv#, Source, AcctsPayCodeCombinationID [Under what Account u r making payment], InvCurrencyCode.
<**AP_INVOICE_LINES_INTERFACE**>: InvoiceID, DistCodeCombinationID, Desc, Amt, CreationDate, CreatedBy, LastUpdateDate, LastUpdatedBy, LineTypeLookupCode.

Concurrent program: Payables Open Interface Import

Validations:
(A) <AP_INVOICES_INTERFACE> - Primary Key – InvoiceID

Interface Columns		**Validations**
Invoice Num (AP_Invoice_All)	Null	Required if there is more than 1 Invoice for the Supplier. Must be unique for supplier
PO Number	Null	Validated against Segment1 <PO_Headers>
Vendor Name	Null	One of the thing is reqd – Vendor ID, NUM or Name.
Vendor Site ID	Null	Validated against VendorSiteID <PO_Vendor_Sites>
PaymentMethodLookupCode	Null	Validated against <AP_Lookup_Code> - Check, Wire, EFT

Base Table	**Interface Col**	**Validations**
APInvoiceAll	InvoiceDate	Value must be valid date format
APInvoiceAll	InvoiceTypeLookupCode	Value must be ‘Standard’ or ‘Credit’ . If InvAmt <0 then “Credit”, else “Standard”
APInvoiceAll	VendorID	Must be existing valid Supplier – vendorID <PO_Vendors>
APInvoiceAll	InvoiceAmt	Value must be equal the um of “Amount” values in AP_Invoice_Lines_Interface for lines with the same InvoiceID.
APInvoiceAll	Source	Lookup value must have Type ‘SOURCE’, otherwise u have to define the source n Payables Lookup Window.
APInvoiceAll	OrgID	

(B) <AP_INVOICE_LINES_INTERFACE>

Interface Columns		Validations
Invoice ID	NN	Validated against <AP_Invoice_Interface>
Receipt Number	NN	To which Invoice ill be match. Validated against <RCV_Shipment_Headers>
Accounting Date	N	Must be valid date format.
PO Number	N	Validated against Segment1 <PO_Headers_All>
Po Distribution ID	N	Used for PO Matching. Validated against <PO_Distribution_All>
Dist Code Combination ID	N	Validated against GL CCID.

Base Table	Interface Col	Validations
AP_ Invoice_ Lines_ Interface	Line Group Number	Value must be positive #
	Amount	Amount = Qty invoiced * Unit Price. If total Amt != Amt of Invoice Header under the same invoiceID, then Import Program will reject the invoice.
	Qty Invoiced	Must be positive for + Amt. And negative for – Amt.
CCID	Balancing Segment	Account code should be valid.
	RCV Transaction ID	

Procedure : Payable Open Interface is used to create invoices from Invoice records in the payable open interface Table. During import payable, we validate invoice Records and rejects invoice records that have insufficient record or data. Successfully imported invoices have distributions and have schedule payments and can be queried, modified and approved in the invoice work bench.

1) We get files from different sources e.g. Flat Files or EDI.
2) We load the data into staging table, then validate the data and then insert the data into AP_Invoice_Interface and AP_Invoice_Line_Interface; And submit to the Payable Open Interface Import Program.
3) If the status = 'Processed' then
   Submit Invoice validation process to validate (Approve) the invoice.
   If the status = 'Rejected' then
   - Fix the invoices in AP_Invoice_Interface
   - Resubmit Payable Open Interface Program.

AR Auto LockBox

Definition: AutoLockbox is a service that commercial bank offers to corporate customers for outsourcing their Accounts Receivables payment processing. We can use AutoLockBox for historical data conversion. We can only load Cash Receipts, not miscellaneous receipts as there is no invoice & customers.

Set Ups done before using Autolockbox Prog:

1) **Set Up Receipt Bank :** We define Bank with **'Account Use': Internal** where checks from customers are deposited. [AR→SetUp→Receipts→Bank]
2) **Set Up Receipt Class** : Here we assign Payment methods. [SetUp→Receipts→Class]
3) **Set Up Receipt Source**: We define Receipt Batch Source and assign Receipt Class, payment method and Bank Account to this source. [Set up→Receipts →Receipt Source]
4) **Define LockBox** : Define Lockbox to use the Receivable AutoLockbox Program. [Set Up→Receipt → LockBoxes →Lockbox]
5) **Define Transmission Format** : Autolockbox uses Transmission format for importing data into receivables. Here we define Lockbox Header, Payment, Lockbox Trailer. [Setup →Receipt→Lockbox→Transmission Format]

[Define LockBox tells how Lockbox will handle Invalid Transaction Number.
1. **Post Partial Amount as Unapplied**: Apply the receipt to the valid transactions, then import the remaining receipt amount with a status of Unapplied.
2. **Reject Entire Receipt**: It doesn't import the invalid receipt and data will remain in Interface table [Ar_Payments_Interface]. We can edit the invalid records in the "**Lockbox Transmission Data Window**", then resubmit the validation step for the receipts before Lockbox can import it into Receivables.
]

Process:
1) We create Control File for flat data file, we get from bank.
2) Move the .dat file and .ctl file to $AR_TOP/bin
3) After that we first do Import, then validation and finally Post QuickCash.

AutoLockBox is a three step Process:-

1. **IMPORT** – During this step, Lockbox reads and formats the data from our bank file into Interface table < AR_PAYMENTS_INTERFACE_ALL> using a SQL*Loader script.
2. **Validation**– This checks data in the interface table for compatibility with Receivables. Once the data is validated , the data is transferred into **QuickCash tables** [Receipt Tables] (AR_INTERIM_CASH_RECEIPTS_ALL, AR_INTERIM_CASH_RCPT_LINES_ALL).
3. **Post Quick Cash -** It applies the receipts and update our customer's balances, So data goes to Base tables.

Interface tables:	AR_PAYMENTS_INTERFACE_ALL
	AR_INTERIM_CASH_RECEIPTS_ALL
	AR_INTERIM_CASH_RCPT_LINES_ALL

Base Tables: AR_CASH_RECEIPTS_ALL,
AR_CASH_RECEIPTS_HISTORY_ALL,
AP_RECEIVABLES_APPLICATION_ALL

Interface table Cols:
For Header: Status, Record Type, LockBox#, Deposit Date, Origination.
For Payment: Status, Record Type, Customer#, Invoice1, Check#, Remittance Amt, Receipt Date, Item Number, LockBox Number.
For Trailer: Status, RecordType, LockBox#, Deposit Date, BatchRecordCount, BatchAmt, Origination.

Validations:

(A) < AR_PAYMENTS_INTERFACE_ALL > **--** [TransmissionRecordID – Pk]

Base Tables	Interface Cols	Validations
	Record Type	Is NN. Type of Record should exist.(Header(HE) / Payment (DE)/ Trailer (TR))
	LockBox Number	Should Exist.
AR_Batches	Deposit Date	Should be there. Entered by user using "Maintain Lockbox Transmission Date".
AR Transmissions (Origin)	Origination	-- do --
AP Bank Branches (Bank Name, Bank Branch Name, Bank #)	Trans Routing#	-- do --
AR Cash Receipt	Receipt Date	-- do --

(B) < AR_INTERIM_CASH_RECEIPTS_ALL > **--** [Cash Receipt ID – Pk]

Base Tables		Validations
Cash receipt ID	NN	Exists
Amount	NN	Should be there.
Currency Code	NN	Should exist in Fnd_Currencies.
GL Date	NN	Should be open.
Receipt Method ID	NN	Payment Method shod be specified.
Remittance Bank Account ID	NN	Shod have Bank Account, Assigned to receipt.
Who's Columns	NN	
Customer Trx ID	N	Should be there
Receipt Number	N	Receipt# - with Cash receipt should be there.

(C) <AR_INTERIM_CASH_RCPT_LINES_ALL>-[Cash_Receipt_ID, Cash_Receipt_Line_ID-Pk]

Base Tables		Validations
Cash Receipt ID	NN	Should exist. Cash Receipt associated with Line.
Cash receipt Line ID	NN	Identifier of the individual, Cash_Receipt_Lines_All
Payment Sequence ID	NN	Should be there
Who's Column	NN	Should be there
Sold to Customer	N	Identified of the customer. Associated with the Interim_Cash_Receipt_Line.
Customer Trx ID	N	Should be there

AR Auto Invoice Interface

SetUp Needed :

1) **Define Transaction [i.e Invoice] Source** : [Set Up→Transaction→Sources]
2) **Define Transaction Flexfield [i.e DFF]** : [Set Up →Financial→Descriptive→Segments

Process:

1) Created Staging Table and then its Synonym in Apps schema.
2) Using SQL*Loader, populate staging table.
3) Run the AutoInvoice Interface (i.e pakage we created) to populate Interface tables.
4) Run the Concurrent Prog : **AutoInvoice Master Program'** for importing to Base Tables.
5) If we get any error, we can use '**AutoInvoice Correct Form**' to fix the errors.

Interface tables: RA_INTERFACE_LINES_ALL,
RA_INTERFACE_DISTRIBUTIONS_ALL
RA_INTERFACE_SALESREPS_ALL

Base tables: **RA_CUSTOMER_TRX_ALL,**
RA_CUSTOMER_TRX_LINES_ALL
RA_CUST_TRX_GL_DIST_ALL
RA_CUSTOMER_TRX_LINE_SALESREPS

Staging Table Col: Customer#, CustomerName, ItemDesc, Ref#, Amt, TrxDate, TrxType, Line#, StatusMsg.

Interface Table Cols:

<RA_INTERFACE_LINES_ALL>
BatchSourceName, SOBID, LineType, Desc, CurrencyCode, Amt, CustTrxTypename, TermName, OrigSystemBillCustomerRef, OrigSystemBillAddressRef, ConversionType,ConversionRate, TrxDate, GLDate, Qty, OrgID, InterfaceLineAttribute 1-4, InterfaceLineContext.

<RA_INTERFACE_DISTRIBUTIONS_ALL>
InterfaceLineAttribute 1-4, Account Class, Org ID, Amount, CCID.

Concurrent Program: Auto Invoice Master Program

Validations: Check for amount, batch source name, conversion rate, conversion type. Validate orig_system_bill_customer_id, orig_system_bill_address_id, validate quantity, Validate if the amount includes tax flag.

(A) <RA_INTERFACE_LINES_ALL>

Base Table <RA Customer Trx All>	**Interface Columns**	**Validations**
<RA Batches All>	Batch Source ID	Must exist in RABatchSourcesAll (Name). BatchSourceType = 'Foreign'.

	Batch Source Name SOB ID, Line Type, Desc, Currency Code, Conversion Type, Trx Date, Receipt Method Name, Interface Status	All are Not Null Columns.
(Exchange Rate)	Conversion Rate	If conversion type = 'user' then this column must not be null, otherwise it must be null.
Exchange Rate Type	Conversion Type	Must exist in <GL_Daily_Conversion_Types>
(Revenue Amount)	Amount	If LineType = 'Charges' Then this col must be NULL.
(Extended Amount)	Amount	When Create_Clearing='No' then AutoInvoice will correct the Revenue Amounts that have the wrong currency precision. When Create_Clearing='No' then It will go to Revenue Amount <RACustomerTrxLinesAll>
	Accounting Rule Duration Accounting Rule ID	

(B) <RA_INTERFACE_DISTRIBUTIONS_ALL> : If in Auto-Invoice, we choose Auto-Accounting, Then we don't need this Interface.

Base Table RA_Cust_Trx_Line_GL_dist_All	Interface Cols	Validations
	Account Class	Must be either Rev, Freight, Tax, Rec, charges, UnBill, or Unearn.
	CCID	Must exist in <GLCodeCombinations>.
RA_Customer_Trx_Lines_All	Interface Line Context	If we pass lines with GlobalContext, then we have to set this col to 'Global Data Elements'
	Percent	The Sum of ll Accunting distribution percentages for a Trx must sum to 100 for an account class.
	Segment 1-6 (If we have 6 Accounting Flexfield Segments)	Valid combination of Accounting Flexfield segment value must exist in <GLCodeCombinations>.

Who's Column :- Last Updated By, Llast Update Date, Created By, Creation Date are Nulls here.

customer API

(Every API has 3 out Parameters – Return_Status, Msg_Count, Msg_Data)

Algorithm Used in API is:

1) We create a record variable of the desired type (Party /Organization)
2) Then we Populate the record with information from source.
3) Then Call the Procedure to create Party / Organization and pass the record to the procedure as a parameter so that procedure put the information of the record variable in the base table.

1. Set the organization id

Exec dbms_application_info.set_client_info('204');

2. Create a party and an account

a) HZ_CUST_ACCOUNT_V2PUB.CUST_ACCOUNT_REC_TYPE
b) HZ_PARTY_V2PUB.ORGANIZATION_REC_TYPE
c) HZ_CUSTOMER_PROFILE_V2PUB.CUSTOMER_PROFILE_REC_TYPE

HZ_CUST_ACCOUNT_V2PUB.**Create_Cust_Account(…)**→ Cust_Account_ID, Account#, PartyID, Party#.

3. Create a physical location

a) HZ_LOCATION_V2PUB.LOCATION_REC_TYPE

HZ_LOCATION_V2PUB.**Create_Location(..)** → Location_ID

4. Create a party site using party_id you get from step 2 and location_id from step 3.

a) HZ_PARTY_SITE_V2PUB.PARTY_SITE_REC_TYPE

HZ_PARTY_SITE_V2PUB.**Create_Party_Site(Party_ID, Location_ID)** →
Party_Site_ID, Party_Site#

5. Create an account site using account_id you get from step 2 and party_site_id from step 4.

a) HZ_CUST_ACCOUNT_SITE_V2PUB.CUST_ACCT_SITE_REC_TYPE

HZ_CUST_ACCOUNT_SITE_V2PUB.**Create_Cust_Acct_Site(Cust_Acct_ID, Party_Site_ID)**
→Cust_Acct_Site_ID

6. Create an account site use using cust_acct_site_id you get from step 5 ans site_use_code = 'BILL_TO'.

a) HZ_CUST_ACCOUNT_SITE_V2PUB.CUST_SITE_USE_REC_TYPE
b) HZ_CUSTOMER_PROFILE_V2PUB.CUSTOMER_PROFILE_REC_TYPE

HZ_CUST_ACCOUNT_SITE_V2PUB.**Create_Cust_Site_Use(Cust_Acct_Site_ID)** →
Site_Use_ID

Interface Table: TCA API.

Base table: HZ_PARTIES HZ_PARTY_SITES HZ_LOCATIONS
HZ_CUST_ACCOUNTS HZ_CUST_SITE_USES_ALL
HZ_CUST_ACCT_SITES_ALL HZ_PARTY_SITE_USES

Validations: Check if legacy values fetched are valid.
Check if customer address site is already created.
Check if customer site use is already created.
Check is customer header is already created.
Check whether the ship_to_site has associated bill_to_site
Check whether associated bill_to_site is created or not.
Profile amounts validation: validate cust_account_id, validate customer status.
Check if the location already exists in HZ_LOCATIONS. If does not exist, create new location.

On-Hand Quantity Interface (Inventory)

Oracle Inventory provide an open interface for us to load transaction from external application and feeder system. These transaction could be sales orders, shipment transaction from an Order Entry System, [or they could be simple material issues, receipts or transfers loaded from data collection devices.]

Interface Table : MTL_Transactions_Interface,
[MTL_Transaction_Lots_Interface,
MTL_Serial_Numbers_Interface]

Base Tables: MTL_On_Hand_Quantities,
[MTL_Lot_Numbers,
MTL_Serial_Numbers]

Staging Table Cols : Org Code, Item No, Source Code, Qty, UOM Code, Sub Inventory, Source HeaderID, Source LineID, Trx Cost, Process Flag, Error Msg,

Interface Table Cols : All Not Null Columns Down

Validations:

1) Valid Organization_Code → Organization_Code <MTL_Parameters>
2) Valid Inventory Item# → Segment1 <MTL_System_Items_B>
3) Valid SubInventory Code → Secondary Inventory Name<MTL_Secondary_Inventories>
4) Valid Transaction UOM → UOM_Code <MTL_Units_Of_Measure>

Interface Columns		Validations
Source code, source lineID, Source HeaderID, who's col, Trx qty, Trx UOM,	NN	
Trx Date	NN	Valid Date Format
Process Flag	NN	1- Ready for Process by Trx Mgr, 2- Not Ready, 3-Error
Transaction Mode	NN	2 - Run Interface Table in Concurrent Prog (submit Manually the Concurrent Prog) 3 – Run in Background Process (occurs Automatically)
Lock Flag	N	1 – Lock, 2 or Null – Not Lock. Should always specify 2.
Organization ID	NN	Should be valid
TRx Type ID	NN	

[The Transaction Mgr picks up the rows to process based on the Process Flag and Transaction Mode to manipulate the records in table.]

PO Requisition Import

Interface tables: PO_REQUISITIONS_INTERFACE_ALL

Base tables: PO_REQUISITIONS_HEADERS_ALL,
PO_REQUISITION_LINES_ALL
PO_REQ_DISTRIBUTIONS_ALL

Columns of Interface Table:

Interface Source Code	NN	Interface Transaction Source
Destination Type Code	NN	Requisition Destination Type
Quantity	NN	Qty Ordered
Authorization Status	NN	Status
Source Type Code	N	Requisition Source
Req Destination ID	N	Req Distribution Unique Identifier

Validations: Check for interface transaction source code,
Check for requisition destination type.
Make sure the currency code exist in Fnd_Currencies.
Check for quantity ordered,
Check for Authorization status type.