A Technical Project

on

Procurement & Sales Management System

Submitted By

Group:

Bindhushree Ragula
Kadiri Neha Reddy
Suhita Pentakota
RVS Pranav Varma

For the internship program conducted by

PramahaSoft Pvt Ltd

Table of Contents

INTRODUCTION

1.1 Overview

In today's digital marketplace, seamless product listings, timely order processing, secure transactions, and transparent buyer-seller interactions are critical to the success of platforms like eBay or OLX. Efficient management of users, listings, orders, and returns significantly enhances customer experience, builds trust, and boosts platform reliability.

The eBay/OLX Management System replicates a dynamic online marketplace environment using Infor ERP LN tools. It enables smooth handling of product uploads, order placements, payment tracking, and return processing. Sellers can easily manage product inventories and descriptions, while buyers can view products, place orders, and track deliveries. The platform ensures consistency and integrity through the use of first-free number logic for auto-generating unique IDs and provides robust support for transactional workflows.

Whether it's a small seller or a high-volume vendor, the system offers flexible and scalable architecture to meet a wide range of needs. With Infor LN's powerful development tools and 3-tier ERP architecture, this system ensures efficient integration of buyers, sellers, and logistics, while supporting real-time updates and financial transparency. Built-in validations and modular design support data accuracy and system maintainability across product, order, and return cycles.

1.2 Scope

The purpose of this documentation is to:

- Provide an overview of how an eBay/OLX-style marketplace can be developed using Infor ERP LN.
- Explain the tools, development processes, procedures, business flow, and validations implemented for managing buyers, sellers, products, orders, payments, and returns.
- Walk through the step-by-step process involved in designing and building an online marketplace system using ERP principles.
- Describe the testing strategies and methodologies used to ensure proper functionality of all modules, including ordering, invoicing, and return handling.
- Present reports and outputs generated during system execution, such as order summaries, invoice records, and user listings.
- Detail the scripts (3GL/4GL) used in the system for automating data management, user actions, and system validations.

1.3 Details

Infor ERP LN provides a powerful and flexible platform to design and implement an eBay/OLX-style online marketplace. By utilizing ERP design principles, this system effectively manages the lifecycle of buyer-seller interactions, from product listings to payments and returns. The system is structured into three primary layers of data: **Master Data, Transactional Data, and Master Master Data**, allowing for streamlined development and maintenance.

I. Master Master Data

These are foundational datasets that are required globally across modules. They include:

- **Countries**: Stores standardized country codes and descriptions used across Buyers and Sellers.
- **Product Groups**: Helps categorize products (e.g., Electronics, Clothing, etc.) for easier filtering and grouping.
- **First Free Number (FFN)**: Manages automatic generation of unique identifiers (IDs) for Buyers, Sellers, Products, Orders, etc.

II. Master Data

These datasets are frequently managed and referenced during transactions. They include:

- **Buyers**: This contains buyer information such as name, contact, and location.
- **Sellers**: Manages seller records including email, address, and business info.
- **Products**: Product listings uploaded by sellers, linked to product groups, with descriptions and other relevant details.

III. Transactional Data

These entities capture the day-to-day transactions and are linked dynamically to the Master Data:

- **Orders**: Tracks individual buyer purchases, linked to the buyer, seller, and product. Includes order status and shipping info.
- **Invoices**: Generated against orders, they summarize payable amounts and taxes.
- **Payments**: Captures payment method, status, and date of transaction. Linked to invoices.
- **Returns**: Handles order return workflows, including status tracking, refund information, and pickup scheduling.

Development Tool

2.1 Development Tools Introduction

Infor LN Development Tools provides tools for developers.

You can use Development Tools to:

- Create Package VRCs in which you can develop software components.
- Set up a data model, consisting of domains and tables, for an application
- Create various types of software components, such as:
  - Sessions
  - Forms
  - Reports
  - Multi-language data field labels
  - Questions and messages
- Create, edit, and compile scripts and libraries, e.g., UI scripts, DLLs, and DALs. The Development Tools functionality is located in the Tools (tt) package on the LN server.

2.2 Overview of software components

The following sections briefly describe various components

- 4GL Engine
- Domains
- Forms
- Functions
- Labels
- Libraries
- Menus
- Messages
- Reports
- Sessions
- SQL queries
- Table definitions
- UI scripts

2.2.1 4GL Engine:

The 4GL engine handles the default behaviour of a session. If additional functionality is needed or default functionality needs to be bypassed, this should be programmed in a UI script (Program script).

When a session is started, the session's object will run. The object and the form work together to perform queries and present data on the form.

The sequence of actions that takes place is:

1. The object allocates memory space for a record buffer based on the table definition.
2. The object builds a SQL statement to select rows from the main table and related tables. The record buffer is populated within a selectdo loop.
3. The record buffer variables are updated after the query and are available for use on forms.
4. The form fields are mapped to the record buffer and displayed on the form.

This sequence of actions is an overview of the actions that take place within a session. The detailed actions are coded. The 4GL Engine (also called the standard program) provides this standardised coding and enforces consistency across sessions in the system.

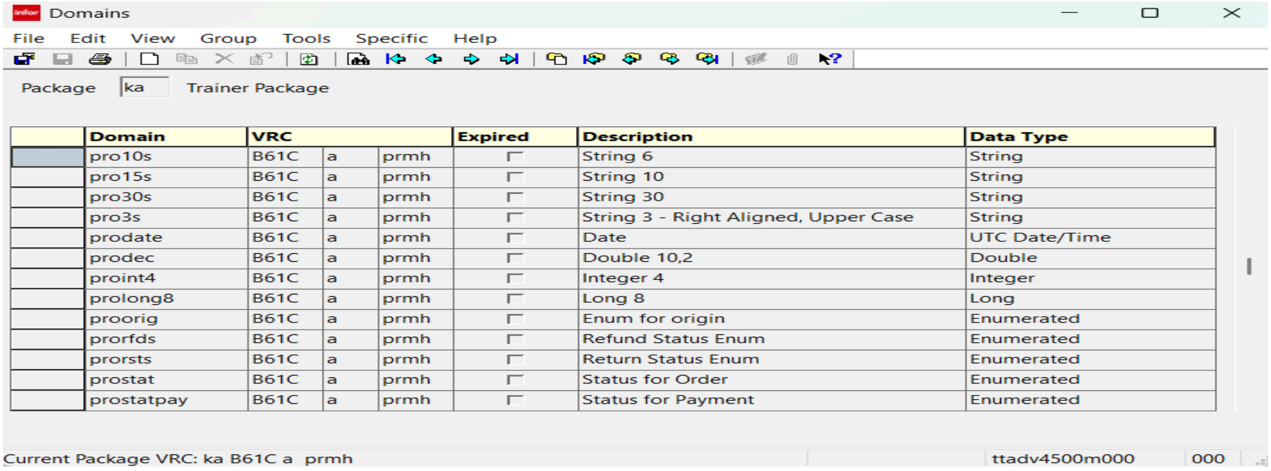
The 4GL Engine controls the processing of a session:

- The session startup includes checking for authorisations, loading icons, and pull-down menus.
- Loading the forms, reports, and charts.
- Performing queries on the main table for the user's company, which includes selects, inserts, updates, and deletes.
- Scrolling through data selections.
- Checking that a reference to another table exists, such as checking that a Department exists for an Employee.
- Handling the right forms, labels, and menus for a user's language. Choose the output device.

2.2.2Domains:

Domains define common information about data, such as data type, length, alignment, valid ranges, display format, and capitalization rules.

Domains ensure consistent data types for fields and variables. Domains can be linked to table fields, form fields, and program variables. The below Figure shows some domains present in the ka (common) package.



Package ka Trainer Package

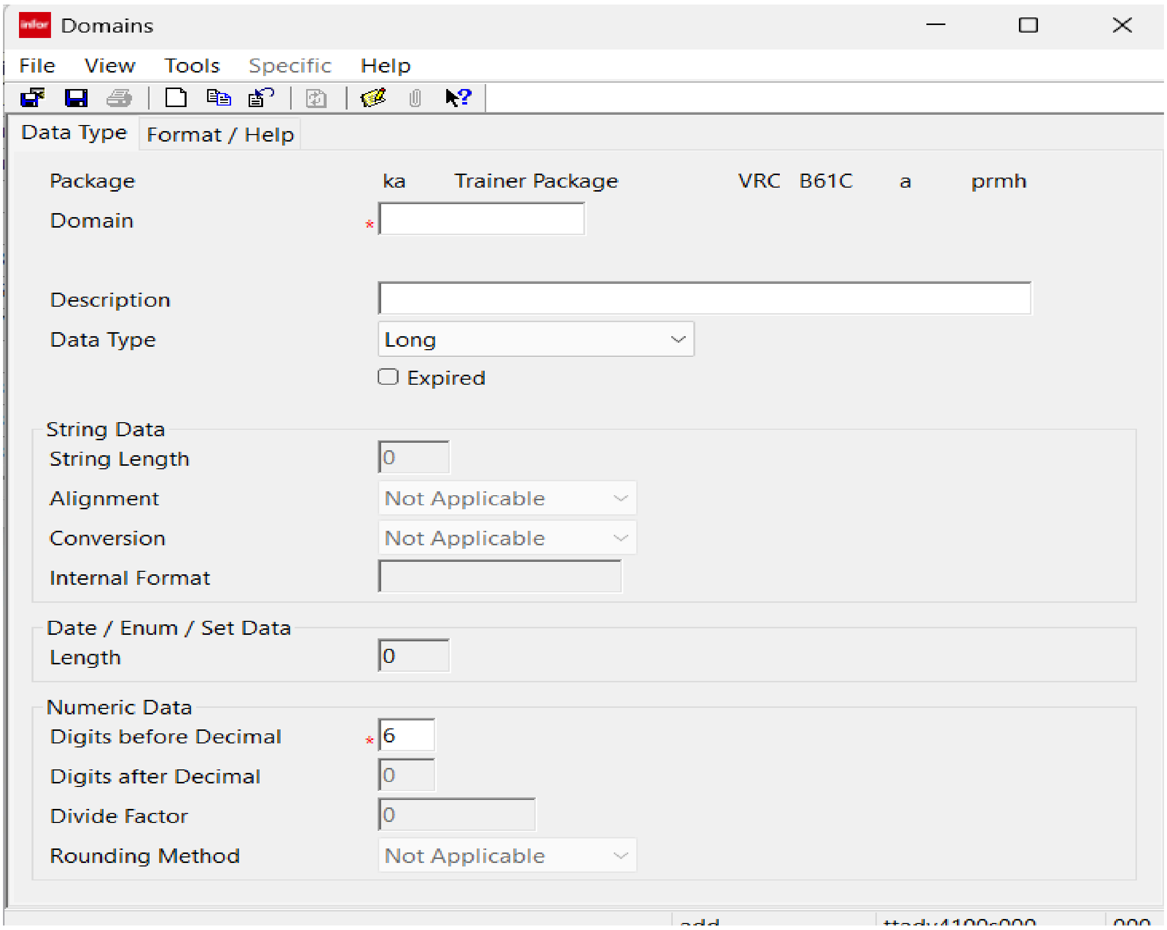
	Domain	VRC			Expired	Description	Data Type
	pro10s	B61C	a	prmh	☐	String 6	String
	pro15s	B61C	a	prmh	☐	String 10	String
	pro30s	B61C	a	prmh	☐	String 30	String
	pro3s	B61C	a	prmh	☐	String 3 - Right Aligned, Upper Case	String
	prodate	B61C	a	prmh	☐	Date	UTC Date/Time
	prodec	B61C	a	prmh	☐	Double 10,2	Double
	proint4	B61C	a	prmh	☐	Integer 4	Integer
	prolong8	B61C	a	prmh	☐	Long 8	Long
	proorig	B61C	a	prmh	☐	Enum for origin	Enumerated
	prorfds	B61C	a	prmh	☐	Refund Status Enum	Enumerated
	prorsts	B61C	a	prmh	☐	Return Status Enum	Enumerated
	prostat	B61C	a	prmh	☐	Status for Order	Enumerated
	prostatpay	B61C	a	prmh	☐	Status for Payment	Enumerated

Current Package VRC: ka B61C a prmh ttadv4500m000 000

Domains of type Enumerated or Set have constants with language-dependent descriptions. Each constant has three characteristics:

- A numeric value that is stored in the database
- Constant Name that the programmer can use
- Constant description that describes the option to the user, in the language of the user.

Creation of a domain can be done in the **Domains(ttadv4500s000)** session, as shown in the figure below



Package ka Trainer Package VRC B61C a prmh

Domain *

Description

Data Type Long

□ Expired

String Data

String Length 0

Alignment Not Applicable

Conversion Not Applicable

Internal Format

Date / Enum / Set Data

Length 0

Numeric Data

Digits before Decimal * 6

Digits after Decimal 0

Divide Factor 0

Rounding Method Not Applicable

2.2.3 Forms:

The form is the user interface part of the session. Forms, which are presented to users, include data and actions that users can perform on that data. The session and form are integrated; one form per session is defined. The form definition in the session identifies the fields, labels, and options that are available in the session's overview display window and details window.

A Form provides the rules for the user dialog, or panel, that is displayed. You create and edit ses sions and forms using the Sessions (ttadv2500s000) session. This session acts as a Developer's workbench as it provides access to all the major components of a session.

To create Forms : To create and edit forms, you must use the Infor LN Dynamic Form Editor that is part of Development Tools. This is a PC based application that should be installed for each devel oper. The Infor LN Dynamic Form Editor also provides access to the major components of the ses sion.

The form editor allows you to specify the available session types for the form (overview and detail), the field groups, labels and fields that are used to create the form.

The form that is presented to the user is created dynamically:

- The form layout is used to determine the labels and fields, the grouping of the fields, and the se quence of the fields.

- The form layout is used to determine the labels and fields, the grouping of the fields, and the se quence of the fields.

- The mode the session is started in determines if the session is in overview mode or detailed mode.

2.2.4 Functions:

Functions allow you to perform a programming task multiple times with different values. A function is declared in the functions section of a script, in a library, or in a separate function script (include).

The possibilities for calling a function depend on how the function is declared:

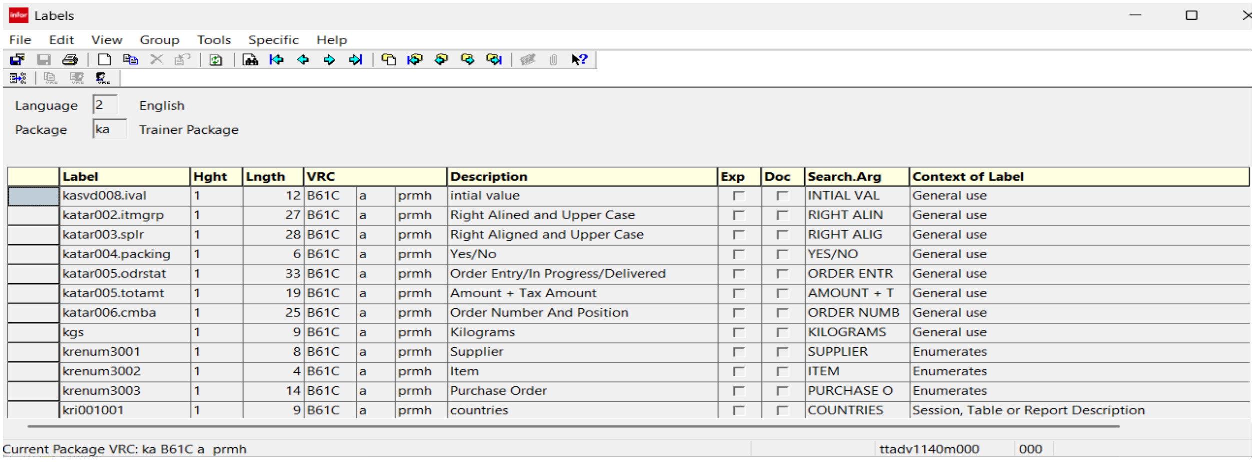
- If a function is declared in the declaration section of a script, you can only call the function within that script.
- If a function is declared in a library, or in a separate function script, you can call the function in multiple scripts and libraries.

To Create Function : To create a function in a program script or in a library, you must edit the script/library via the Program Scripts / Libraries (ttadv2530s000) session. If you define a function in a library, you must link that library to the scripts and libraries where you want to call the function.

To create a function in a separate function script:(include), you must create the function script through the Functions (ttadv2560s000) session. You must include this function script (through an "#include" statement) in the scripts and libraries where you want to call the function.

2.2.5 Labels:

A label is a code that is used instead of language-dependent text in forms, reports, and menus. A la bel consists of a name and a content description. The content of a label can differ by language, but the label name remains the same for all languages.



Language 2 English

Package ka Trainer Package

	Label	Hght	Lngth	VRC			Description	Exp	Doc	Search.Arg	Context of Label
	kasvd008.ival	1	12	B61C	a	prmh	intial value	□	□	INTIAL VAL	General use
	katar002.itmgrp	1	27	B61C	a	prmh	Right Alined and Upper Case	□	□	RIGHT ALIN	General use
	katar003.splr	1	28	B61C	a	prmh	Right Aligned and Upper Case	□	□	RIGHT ALIG	General use
	katar004.packing	1	6	B61C	a	prmh	Yes/No	□	□	YES/NO	General use
	katar005.odrstat	1	33	B61C	a	prmh	Order Entry/In Progress/Delivered	□	□	ORDER ENTR	General use
	katar005.totamt	1	19	B61C	a	prmh	Amount + Tax Amount	□	□	AMOUNT + T	General use
	katar006.cmba	1	25	B61C	a	prmh	Order Number And Position	□	□	ORDER NUMB	General use
	kgs	1	9	B61C	a	prmh	Kilograms	□	□	KILOGRAMS	General use
	krenum3001	1	8	B61C	a	prmh	Supplier	□	□	SUPPLIER	Enumerates
	krenum3002	1	4	B61C	a	prmh	Item	□	□	ITEM	Enumerates
	krenum3003	1	14	B61C	a	prmh	Purchase Order	□	□	PURCHASE O	Enumerates
	kri001001	1	9	B61C	a	prmh	countries	□	□	COUNTRIES	Session, Table or Report Description

Current Package VRC: ka B61C a prmh ttadv1140m000 000

For each label code, you can specify:

- A label description. This is the label text which you can edit and translate.

- The length. This is the number of characters of the description.
- The height. This is the number of lines of the description.

To create labels: You can create labels in the Labels (ttadv1140m000) session. If you create new labels, you must compile these in the Compile Labels (ttadv1243m000) session so that Infor Enterprise Server can display the new labels at run time.

2.2.7 Libraries:

A library, also called Dynamic Link Library (DLL), provides application-specific functions that can be used throughout the system, by many sessions. A library is a script that is stored in a separate component. The library is compiled independently of the program scripts that use it. Libraries are loaded at runtime by sessions that use them. When a session needs to access a library, the library is loaded, and the relevant routine is executed.

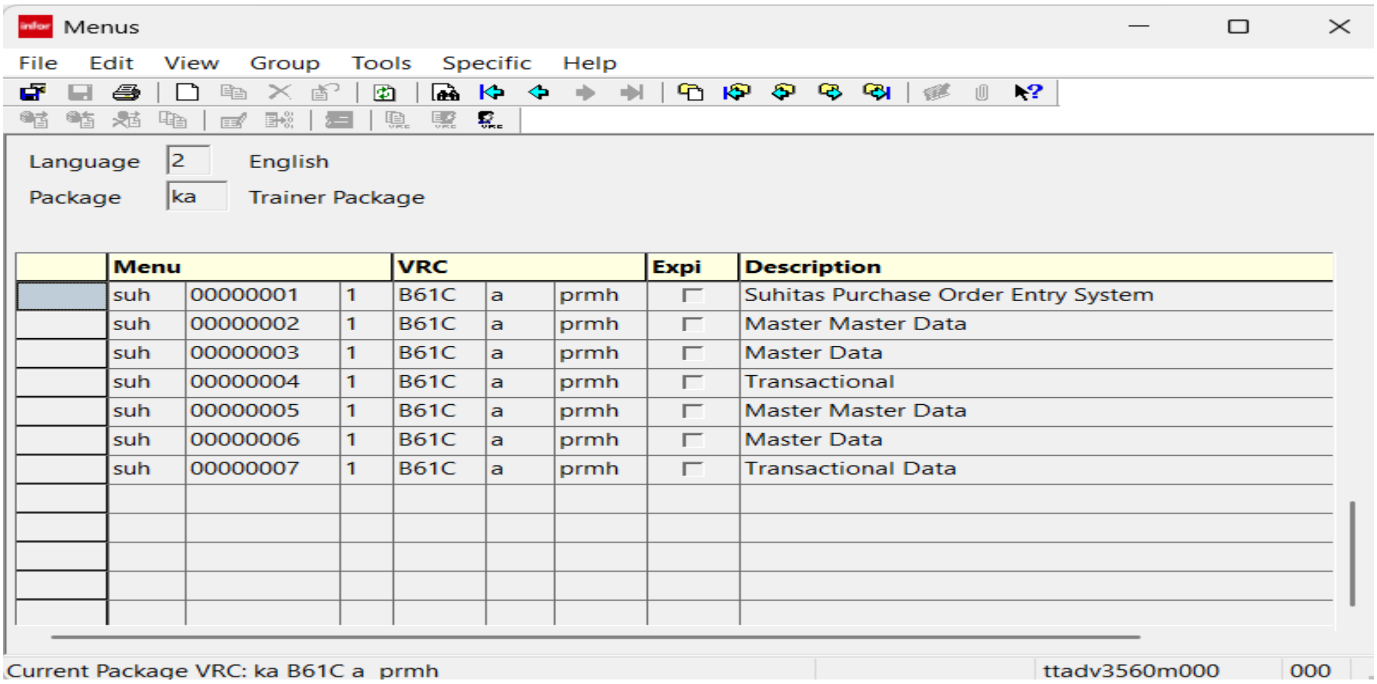
To create libraries: You can create and edit libraries in the Program Scripts / Libraries (ttad v2530m000) session.

Libraries are a type of program script. The program script code should start with "dll" and the script type should be "General Library". This helps categorise the scripts into their general use.

2.2.8 Menus:

Menus are used to organise the Infor LN sessions in a logical folder / subfolder structure. The fold ers and subfolders usually represent Infor LN packages and modules. The Menu browser is used to open the folders and subfolders in order to find the sessions. Sessions can be launched directly from the Menu browser. The Menu browser is used in Infor Ming.le, Infor ES Web UI, and Infor LN Worktop.

Infor LN users can have their own customised menus. The start menu for a user must be defined in the User Data (ttaad2500m000) session.

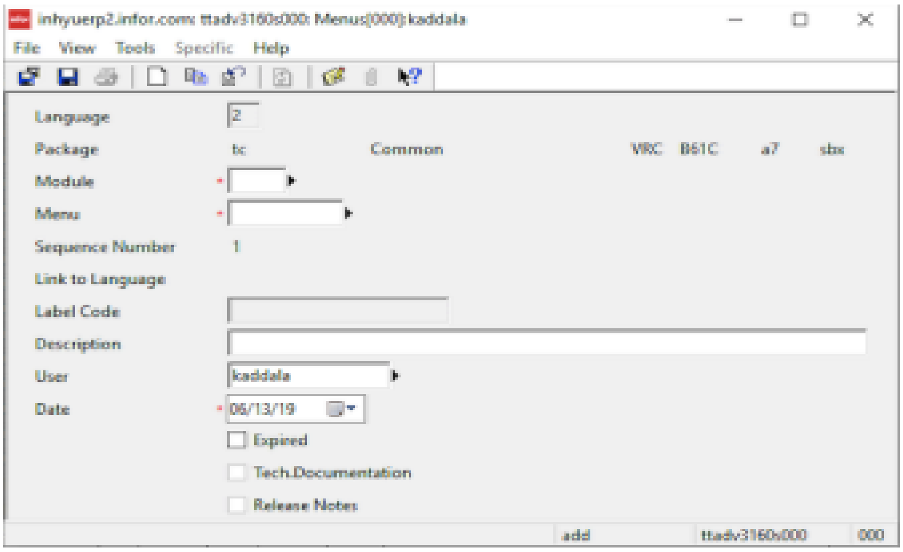


Language 2 English
Package ka Trainer Package

Menu			VRC			Expi	Description
suh	00000001	1	B61C	a	prmh	□	Suhitas Purchase Order Entry System
suh	00000002	1	B61C	a	prmh	□	Master Master Data
suh	00000003	1	B61C	a	prmh	□	Master Data
suh	00000004	1	B61C	a	prmh	□	Transactional
suh	00000005	1	B61C	a	prmh	□	Master Master Data
suh	00000006	1	B61C	a	prmh	□	Master Data
suh	00000007	1	B61C	a	prmh	□	Transactional Data

Current Package VRC: ka B61C a prmh ttadv3560m000 000

To create menus: To create menus you must use the Menus (ttadv3560s000) and Menu Fields (ttadv3560s000) sessions.


inhyuerp2.infor.com: ttadv3160s000: Menus[000]:kaddala

Language	2				
Package	tc	Common	VRC B61C	a7	sbx
Module	*				
Menu	*				
Sequence Number	1				
Link to Language					
Label Code					
Description					
User	kaddala				
Date	* 06/13/19				

□ Expired

□ Tech.Documentation

□ Release Notes

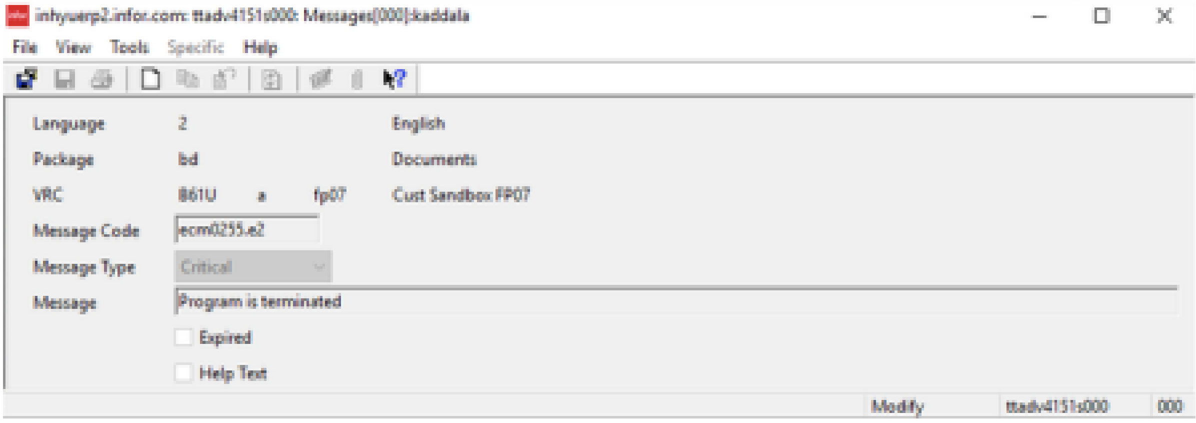
add ttadv3160s000 000

2.2.9 Messages:

Messages are language-independent software components that allow you to customize dialog messages.

For each message, you can specify:

- The message code. This is the unique identifier of the message across all languages.
- The message type, e.g. “Warning".
- The message. This is the message text which you can edit and translate. This message can contain codes that are substituted when the message is displayed.


inhyuerp2.infor.com: ttadv4151s000: Messages[000]:kaddala

Language	2			English
Package	bd			Documents
VRC	B61U	a	fp07	Cust Sandbox FP07
Message Code	ecm0255.e2			
Message Type	Critical			
Message	Program is terminated			

□ Expired

□ Help Text

Modify ttadv4151s000 000

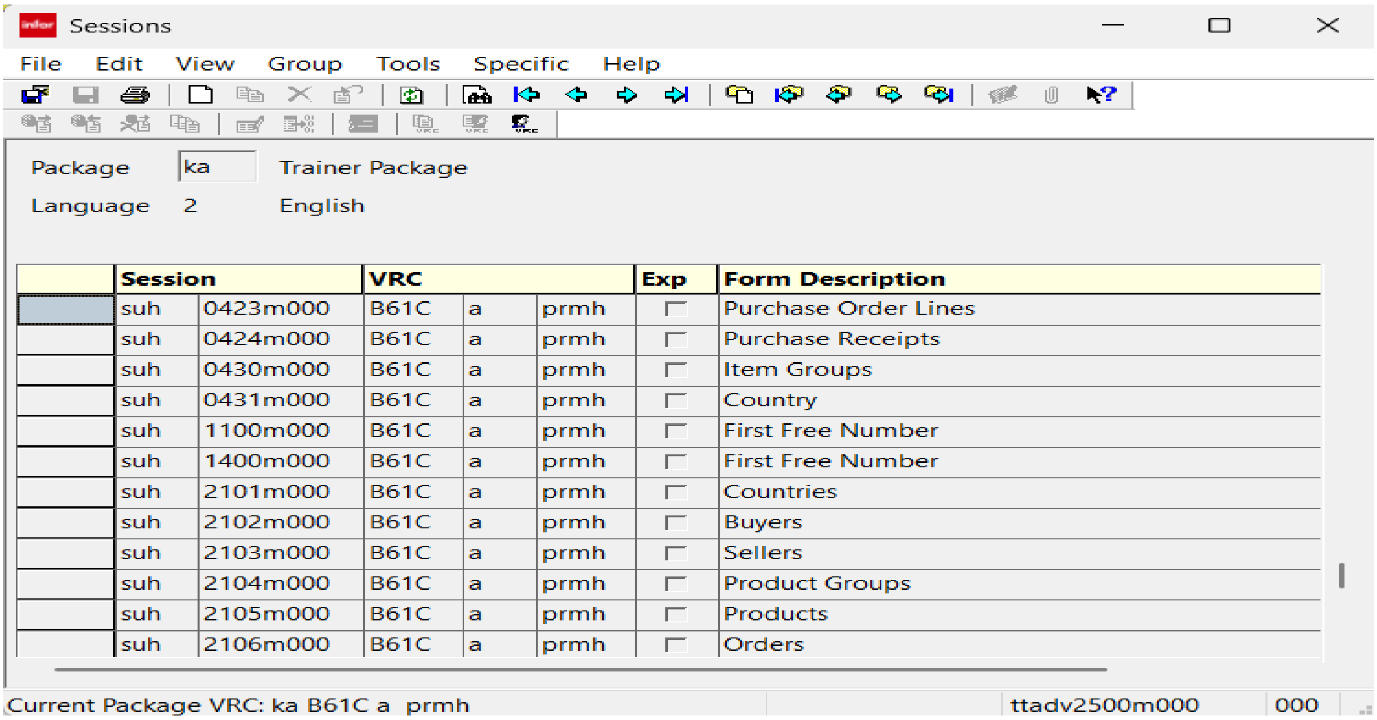
To create messages: You can create messages in the Messages (ttadv4551m000) session.

2.2.10 Sessions:

A session performs an activity. Sessions are used to present data, edit data, and process data. Each session has a code. The session code is displayed in the status bar of the session window. A session consists of multiple components that work together, such as a form and an object. A session object is a compiled UI script.

Overview session:

An overview session shows multiple rows from a table in a grid. Overview sessions are also called Multi-Occurrence sessions, because they display multiple records, or occurrences, from the same table. Scroll bars allow you to view rows above or below the current rows that are displayed in the grid. Based on security authorisations, you are able to insert, edit, copy, and delete entries in the grid.



Package ka Trainer Package

Language 2 English

	Session		VRC			Exp	Form Description
	suh	0423m000	B61C	a	prmh	□	Purchase Order Lines
	suh	0424m000	B61C	a	prmh	□	Purchase Receipts
	suh	0430m000	B61C	a	prmh	□	Item Groups
	suh	0431m000	B61C	a	prmh	□	Country
	suh	1100m000	B61C	a	prmh	□	First Free Number
	suh	1400m000	B61C	a	prmh	□	First Free Number
	suh	2101m000	B61C	a	prmh	□	Countries
	suh	2102m000	B61C	a	prmh	□	Buyers
	suh	2103m000	B61C	a	prmh	□	Sellers
	suh	2104m000	B61C	a	prmh	□	Product Groups
	suh	2105m000	B61C	a	prmh	□	Products
	suh	2106m000	B61C	a	prmh	□	Orders

Current Package VRC: ka B61C a prmh ttadv2500m000 000

A details session shows one row and is also called a single-occurrence session. The details session is opened when you insert, edit or copy from the overview session. The details session allows you to edit individual field values and save your changes. The session may also have additional options that can be performed using buttons, icons or specific menu options.

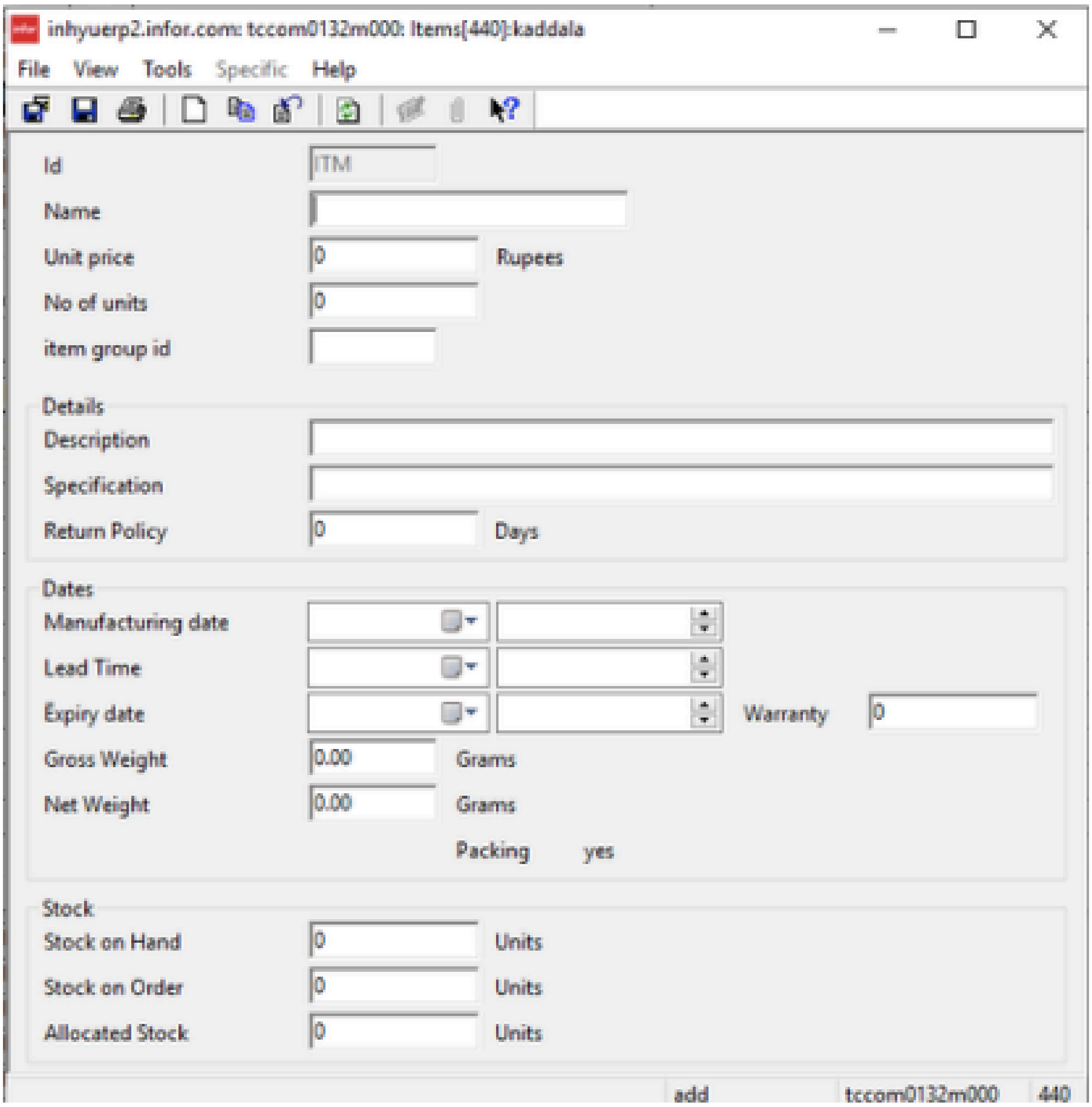
Synchronization:

The details session is a synchronized dialog of the overview session. Synchronized sessions work together.

- When the details record changes, the overview session will show the changes if the affected fields are on the overview session form.
- When a record is selected in the overview session, the details session will show the selected record.

Details session:

A detailed session shows one row and is also called a single-occurrence session. The details session is opened when you insert, edit or copy from the overview session. The details session allows you to edit individual field values and save your changes. The session may also have additional options that can be performed using buttons, icons or specific menu options.



Id	ITM	
Name		
Unit price	0	Rupees
No of units	0	
item group id		

Details

Description		
Specification		
Return Policy	0	Days

Dates

Manufacturing date				
Lead Time				
Expiry date			Warranty	0
Gross Weight	0.00	Grams		
Net Weight	0.00	Grams		
	Packing	yes		

Stock

Stock on Hand	0	Units
Stock on Order	0	Units
Allocated Stock	0	Units

Usually the overview session and the details session are two separate sessions.However, the over view session and the details session can be the same session.You use the form editor to indicate the fields that appear in the overview session, and which fields appear in the details session.

2.2.11 SQL Queries:

You can create and edit sessions and forms using the Sessions (ttadv2500m000) session. This session acts as a Developer's workbench as it provides access to all the major components of a session.

Infor LN supports multiple Relational Database Management Systems. Each database system is supported in the architecture by using a driver: a database specific program that translates SQL syntax from the SQL used in the object to the SQL used by the database system.

4GL Program queries:

You can create queries in 4GL scripts or libraries, e.g. in a UI script, a report script, a DLL or a DAL. These queries can read and update data in the database. At runtime, the queries are triggered during session execution, for example by an event or by a form command.

An embedded query starts with a SELECT statement that allows you retrieve a selection of data from a number of tables based on conditions defined in the WHERE clause.

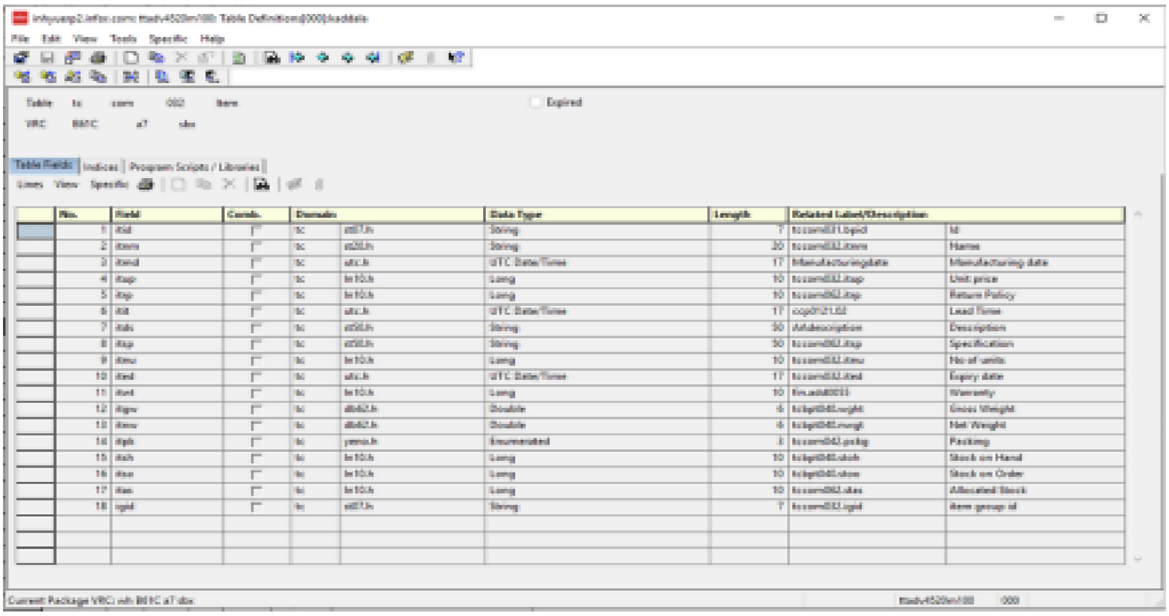
An embedded query starts with a SELECT statement that allows you retrieve a selection of data from a number of tables based on conditions defined in the WHERE clause.

```
SELECT columns ...
FROM table(s) ...
WHERE each row fulfills the condition(s) ...
```

The columns you select are typically table fields that are selected from tables. The where clause in an embedded query allows you to specify conditions using either columns that are selected or based on variables declared in your program script. The rows you select can be processed individually within a selectdo loop. The selectdo loop provides an iteration mechanism for the records selected by the SELECT statement.

2.2.12 Table definitions:

A table definition defines the structure of a table. A table definition contains fields and indices. Table fields are linked to domains that define the data type and several characteristics of the fields.



File Edit View Tools Specific Help
Table Fields | Indices | Program Scripts / Libraries
Lines View Specific
Expired

Table tc com 032 Item
VRC B61C a7 dev

No.	Field	Comb.	Domain		Data Type	Length	Related Label/Description	
1	itid	□	tc	st07.h	String	7	tccom031.bpid	Id
2	itnm	□	tc	st20.h	String	20	tccom032.itnm	Name
3	itmd	□	tc	utc.h	UTC Date/Time	17	Manufacturingdate	Manufacturing date
4	itup	□	tc	lm10.h	Long	10	tccom032.itup	Unit price
5	itrp	□	tc	lm10.h	Long	10	tccom062.itrp	Return Policy
6	itlt	□	tc	utc.h	UTC Date/Time	17	ccp0121.02	Lead Time
7	itds	□	tc	st50.h	String	50	Atdescription	Description
8	itsp	□	tc	st50.h	String	50	tccom062.itsp	Specification
9	itnu	□	tc	lm10.h	Long	10	tccom032.itnu	No of units
10	ited	□	tc	utc.h	UTC Date/Time	17	tccom032.ited	Expiry date
11	itwt	□	tc	lm10.h	Long	10	tinadd0031	Warranty
12	itgw	□	tc	db62.h	Double	6	tcbpt040.ugwt	Gross Weight
13	itnw	□	tc	db62.h	Double	6	tcbpt040.nwgt	Net Weight
14	itpk	□	tc	yesno.h	Enumerated	8	tccom042.pckg	Packing
15	itsh	□	tc	lm10.h	Long	10	tcbpt040.stoh	Stock on Hand
16	itso	□	tc	lm10.h	Long	10	tcbpt040.stoo	Stock on Order
17	itas	□	tc	lm10.h	Long	10	tccom062.itas	Allocated Stock
18	itgid	□	tc	st07.h	String	7	tccom032.igid	Item group id

Current Package VRC: wh B61C a7 dev ttadv4520m100 000

Fields, domains and indices:

A table has fields. Table fields store individual pieces of data such as a customer name, the quantity of an item ordered, or the date that a journal entry was made.

Table fields are linked to domains. Domains are components that define common information about data such as:

- the data type, e.g. a character type for customer name, a number type for quantity, and a date type for journal date.
- Valid ranges.
- Special characters such as capitalization rules.

A table must have at least one index. An index consists of one or more table fields that are used to sort and search records in the table. The first index is always the Primary key, which is the unique identification for a record in a table.

Related Tables and References:

The table may have a related table. A related table means that a field in the table will refer to the key field of another table. In this way, data can have relationships: customers can have orders; inventory stores items in warehouses; and employees work in a department.

To create table definitions: You can create table definitions in the Table Definitions (ttad v4520m000) session.

2.2.13 UI Scripts:

The default behaviour of a session is handled by the 4GL engine. If you require additional functionality or want to bypass the default functionality, you program your changes in the session's UI script (Program script). The UI script is compiled in the session object. The object contains only the ex ceptions to the normal operating procedures of the system. The 4GL Engine executes the normal operating procedures of the system, and you write the exceptions.

The UI script contains events to hook into the 4GL Engine, and describe the actions that should happen as a result of an event. Events trigger code. When the session runs and an event triggers, the 4GL Engine uses the session object to perform the appropriate actions.

Actions that could take place as the result of an event are:

- Calculate a field value before a field displays.
- Synchronize the current (parent) session with a child session when you run a form command. For example: a parent session that uses the Orders table is synchronized with a child session that uses the Order lines table.
- Send a file to the user’s PC when you save a record

The syntax of the UI script contains complete programming syntax capabilities such as variable de clarations, expressions, operators, transfer of control, iterations (loops), functions, embedded SQL, and a full featured functions library.

To create UI Scripts: You can create and edit UI scripts in the Program Scripts / Libraries (ttad v2530m000) session. When you create a new session in the Sessions (ttadv2500m000) session, a corresponding UI script is generated automatically.

Development Parameters

To maintain or create software components, a developer requires default development settings and parameters and authorization to at least one package VRC.

Development Parameters:

The settings and parameters a developer needs, are defined in a Development Parameters template. Parameters are available for the following:

- Automatic compilation to the run-time data dictionary after changes to forms or menus
- Actions to be performed automatically after the Copy to Current Package VRC option
- The parameters that the editor can use to develop software

3.1 Development Authorizations:

A developer needs authorization for at least one package VRC.

To define developer authorizations for a user, you can do one of the

following: • Link the user to a Developer Authorization template.

- Give authorization for all package VRCs.

3.2 System Requirements

System Requirements play a vital role in the development life cycle (SDLC). Any changes made to the requirements in the future will have to go through a formal change approval process. • Tool : INFOR ERP LN (version 7.6)

- Database : Microsoft SQL Server 2000
- Operating system: Linux

3.3 Hardware Requirements

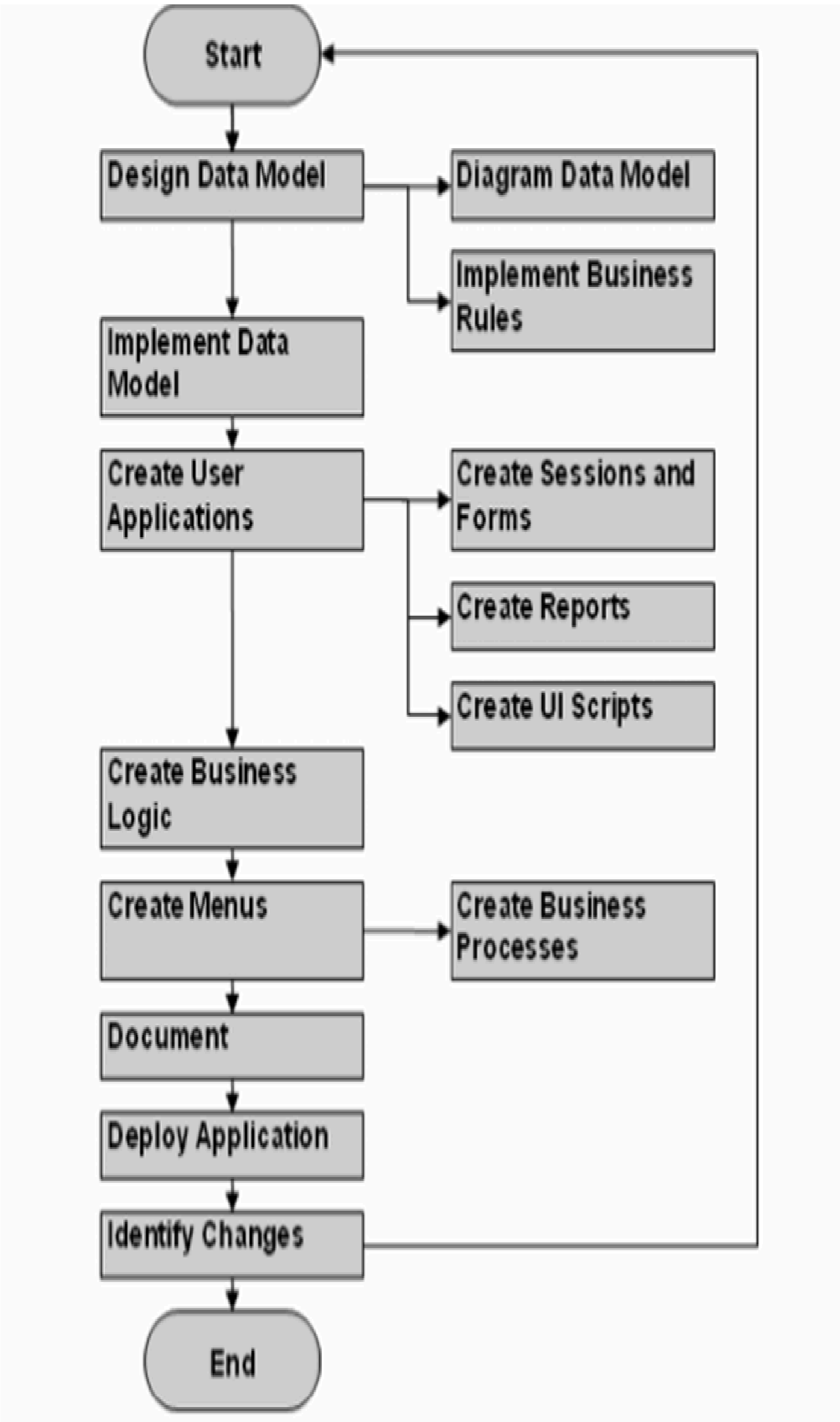
- Ram : 4 GB (Minimum)
- Processor : Core 2 Dual
- Hard Disk : 40 GB free space (Minimum)

Development Procedures

4.1 Development Process

Developers use the Infor LN Development Tools to develop the software components that make up the application.

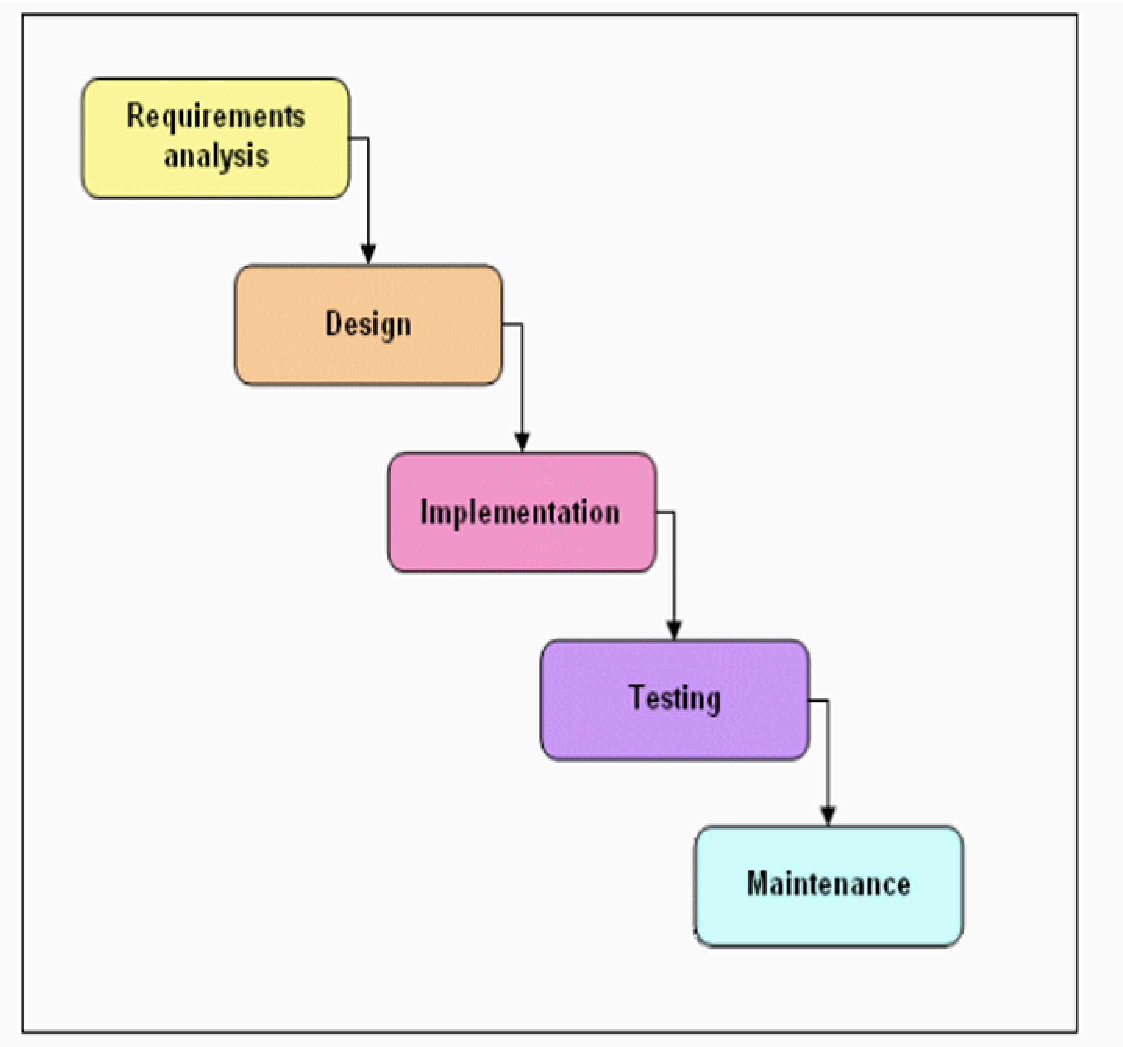
There is a logical progression in developing the components to create an application. The develop ment process explains this progression.



4.2 Development Life Cycle

The software development life cycle consists of Water Flow Data Model

Different Stages present in the Water flow data model are as follows



Requirement gathering:

Requirements describe the "what" of a system. In requirements analysis phase, the requirements are properly defined and noted down. The output of this phase is SRS (Software Requirements Specification) document written in natural language.

Design:

In this phase, a logical system is built which fulfils the given requirements. Design phase of software development deals with transforming the customer's requirements into a logically working system. Here we have two steps:

Primary Design Phase: In this phase, the system is designed at block level.

Secondary Design Phase: In this phase, detailed design of every block is performed.

Implementation:

In this phase, the design document is coded according to the module specification. This phase transforms the SDD document into a high level language code. Once a module is developed, a check is carried out to ensure that coding standards are followed.

Testing:

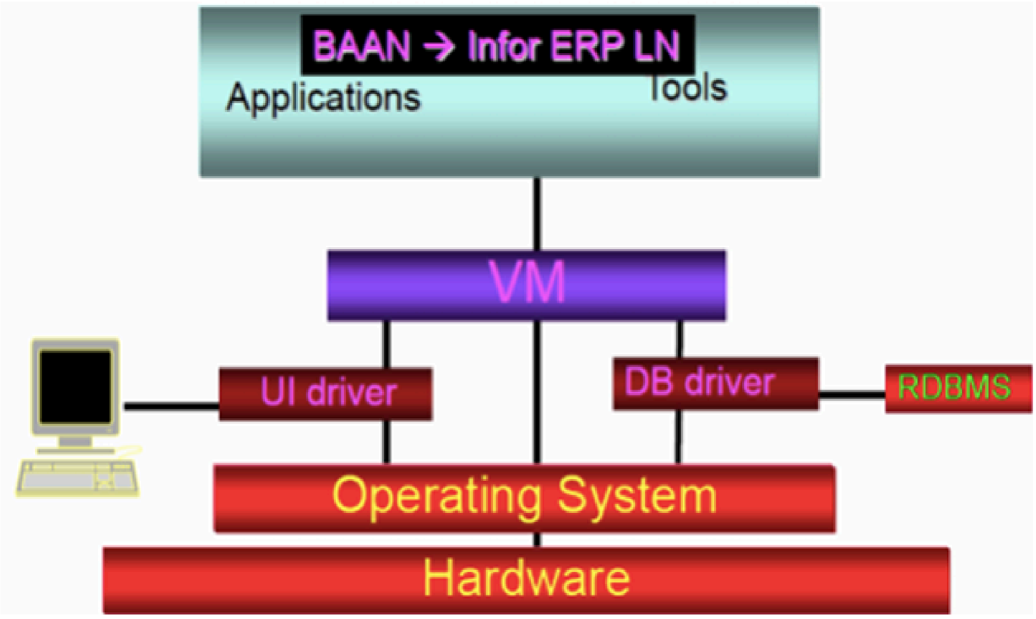
Testing is a process of running the software on manually created inputs with the intention to find errors. In process of testing, an attempt is made to detect errors, to correct the errors in order to develop error free software.

Maintenance:

Software Maintenance is done to rectify the errors which are encountered during the operation of software and to change the program function to interface with new hardware or software to change the program according to increased requirements.

4.3 Architecture

Infor ERP supports a three-tier architecture consisting of a user interface tier, an application tier, and a database tier. The user interface tier provides presentation and input services for user interaction. The application tier consists of the Infor ERP application server and the application programs. The database tier includes the Infor ERP database driver and a third party RDBMS product that acts as the database server.



The database driver is the interface between the Infor ERP applications and the RDBMS server. The database driver translates database requests from the Infor ERP application server to RDBMS specific SQL requests that it sends to the database server. After the database server retrieves the requested information, the database driver then passes the data back to the Infor ERP application server.

User Interface Tier:

The user interface tier consists of the Infor ERP user interface for Microsoft Windows (called 'BW') and for Internet browsers (called 'BI'). Data input from the user through BW or BI is relayed to the Infor ERP application server; data returned from the Infor ERP application server is displayed to the user in graphical form by the user interface.

Application Tier:

The application tier includes both the application programs and the Infor ERP application server. Together, the application programs and the application server provide much of the functionality of Infor ERP. The application programs include the compiled Infor ERP applications and the data dictionary. The Infor ERP applications are written in the 4GL programming language with embedded SQL using the development environment provided by the Infor ERP Tools package.

The Infor ERP application server schedules and runs the application programs, sends and receives information to and from the user interface server, and initiates an instance of the database driver as necessary for communication with the database server. A running database driver can support multiple connections to a single RDBMS instance. If a Infor ERP installation stores data tables in multiple databases, the application server must start one instance of the database driver for each database with which it must communicate. The Infor ERP application server has traditionally been called `Baan shell' or simply `bshell'.

Database Tier:

The database tier consists of the Infor ERP database driver and the database server. The database driver provides a common interface between the Infor ERP application server and the database server. Communication between the application server and the database driver is the same, no matter which RDBMS product is used as the database server. There is one database driver for each of the RDBMS products that Infor ERP supports. Communication between the database driver and the database server is tailored to the RDBMS being used. The database driver communicates with the RDBMS through structured query language (SQL) statements and the native application programming interface (API) of the RDBMS.

The database server consists of one of five third party RDBMS products: Oracle, Informix, Sybase, DB2, or Microsoft SQL Server. All Infor ERP application data is stored in a relational database managed by an RDBMS. It is possible to have multiple RDBMS products in one Infor ERP installation, with some data residing in one database server and other data residing in another.

4.4 Business Flow

In the eBay/OLX Management System, each order is raised by a registered buyer against a listed product uploaded by a seller. The transaction flows through several interconnected tables, managed using Infor LN sessions.

1. Selling Workflow

- **Seller Registration:**
  Seller provides business details which are inserted into the `seller` table. The selected country is validated using a foreign key reference to `country.country`.

- **Product Grouping:**
  Product categories are defined and maintained in the `productgroup` table. A group can be shared across multiple products and sellers.

- **Product Upload:**
  Sellers upload products which are stored in the `product` table. Product ID (`pid`) is auto-generated using the `firstfreenumber` table with origin code `PRODUCT`.
  Foreign key constraints:

  - `sid` → `seller.sid`

  - `productgpID` → `productgroup.productgpID`

- **Product Activation:**
  Once uploaded, the product becomes available for buyer selection in the marketplace.

2. Buying Workflow

- **Buyer Registration:**
  Buyers register and their information is stored in the `buyer` table, validated via `country`.

- **Placing an Order:**
  Buyers select a product and place an order.

  - `orderID` is generated using `firstfreenumber` (origin = `ORDERS`)

  - A new row is inserted into the `orders` table with default status `placed`.
    Foreign key validations:

  - `bid` → `buyer.bid`

  - `sid` → `seller.sid`

  - `pid` → `product.pid`

- **Order Lifecycle:**
  Status transitions for an order:

  - `placed` → `dispatched` → `delivered`

  - If returned: `return in progress` → `returned` (if accepted)

3. Billing & Return Workflow

A. Invoice Generation:

- Triggered when order status = `placed`

- `invoiceID` is auto-generated via `firstfreenumber` (origin = `INVOICE`)

- Invoice inserted into `invoice` table

  - Links: `orderID`, `buyerID`

  - Amount and tax computed and stored

B. Payment Processing:

- Triggered after invoice generation
- `paymentID` is generated using `firstfreenumber` (origin = `PAYMENT`)
- Entry is added in `payment` table
  - Foreign key: `invoiceID`
  - Status values: `paid`, `pending`, `failed`

C. Return Handling:

- Triggered when buyer initiates return
- Entry inserted into `return` table
  - Linked to `orderID`
  - Includes refund status, pickup date, and refund date
- Order status transitions to:
  - `return in progress` → `returned`
- Refund finalization updates refund status and refund date

4. First Free Number Management

- Purpose: To generate unique IDs for `order`, `invoice`, `product`, `payment`, `return`.
- Managed via the `firstfreenumber` table.
  - Example origins: `ORDERS`, `INVOICE`, `PAYMENT`, `PRODUCT`, `RETURN`
- After each insert, the corresponding `ffnu` is incremented to ensure the next free number is ready.

4.5 Validations

Products:
- Product ID must be auto-generated using First Free Number (`PRODUCT`).
- Seller ID must exist in the `seller` table.
- Product Group ID must exist in `productgroup` table.
- All fields (description, price, group) must be validated before product is marked active.

Buyers & Sellers:
- Country code must exist in `country` table.
- Email, contact number, and name must be provided and verified.

Orders:

- Order date should be set to current date and disabled for manual edit.
- Order ID must be auto-generated using First Free Number (`ORDERS`).
- Quantity must be greater than zero.
- Order status should default to `placed`.
- FK validations: buyer ID, seller ID, and product ID must exist.

Invoices:

- Invoice ID must be generated using First Free Number (`INVOICE`).
- Total and tax amounts should be automatically calculated and stored.
- Invoice must be linked to a valid order ID and buyer ID.

Payments:

- Payment ID should be generated via First Free Number (`PAYMENT`).
- Payment status must be among: `paid`, `pending`, `failed`.
- Payment date must be set to current date and be uneditable.

Returns:

- Return ID should be generated using First Free Number (`RETURN`).
- Refund status (`refunded`, `not refunded`) and return status (`in progress`, `returned`) must be set correctly.
- Refund date and pickup date must be recorded.
- Return can only be created if the related order status is `delivered`

System Designs

System Design:

System design is a process through which requirements are translated into a representation of soft ware. Initially the representation depicts a holistic view of software. Subsequent refinement leads to a design representation that is very close to source code. Design is a place where quality fostered in software development. Design provides us with representation of software that can be assessed for quality; this is the only way that can accurately translate the customer requirements into finished software product or system. System design serves as the foundation for all engineering and software maintenance steps that follow, we look the design process from three distinct perspectives. • Conceptual Design

- Logical Design
- Physical Design

The higher view is the conceptual view, followed by the logical view and finally the physical view. In designing application, we generally begin and end each phase in a sequentially order, although they may overlap one another along the way.

5.1 Data Models

In order to create a Purchase Order entry system we need

I. Master Master Data

These are foundational datasets that are required globally across modules. They include:

- **Countries**: Stores standardized country codes and descriptions used across Buyers and Sellers.
- **Product Groups**: Helps categorize products (e.g., Electronics, Clothing, etc.) for easier filtering and grouping.
- **First Free Number (FFN)**: Manages automatic generation of unique identifiers (IDs) for Buyers, Sellers, Products, Orders, etc.

II. Master Data

These datasets are managed and referred to frequently during transactions. They include:

- **Buyers**: Contains buyer information such as name, contact, and location.
- **Sellers**: Manages seller records including email, address, and business info.
- **Products**: Product listings uploaded by sellers, linked to product groups, with descriptions and other relevant details.

III. Transactional Data

These entities capture the day-to-day transactions and are linked dynamically to the Master Data:

- **Orders**: Tracks individual buyer purchases, linked to the buyer, seller, and product. Includes order status and shipping info.

- **Invoices**: Generated against orders, they summarize payable amounts and taxes.

- **Payments**: Captures payment method, status, and date of transaction. Linked to invoices.

- **Returns**: Handles order return workflows, including status tracking, refund information, and pickup scheduling.

5.2 Procurement & Sales Data Model

1. Buyer

Field Name	Data Type	Size	Description	Key
BID	string	10	Buyer ID	PK
Name	string	50	Full Name	
Phno	string	15	Phone Number	
Address	string	100	Residential Address	
Country	string	3	Country Code	FK → Country.country

2. Seller

Field Name	Data Type	Size	Description	Key
SID	string	10	Seller ID	PK
Name	string	50	Full Name	
Phno	string	15	Phone Number	
Address	string	100	Business Address	
Country	string	3	Country Code	FK → Country.country
Email	string	50	Contact Email	

3. Country

Field Name	Data Type	Size	Description	Key
country	string	3	ISO Country Code	PK
desc	string	50	Country Name	

4. Product

Field Name	Data Type	Size	Description	Key
PID	string	10	Product ID	PK
SID	string	10	Seller ID	FK → Seller.SID

productgpID	string	10	Product Group ID	FK → ProductGroup.productgpID
desc	string	100	Product Description	

5. Product Group

Field Name	Data Type	Size	Description	Key
product group ID	string	10	Product Group ID	PK
desc	string	50	Group Description	

6. Orders

Field Name	Data Type	Size	Description	Key
orderID	string	10	Order ID	PK
SID	string	10	Seller ID	FK → Seller.SID
BID	string	10	Buyer ID	FK → Buyer.BID
PID	string	10	Product ID	FK → Product.PID
quantity	integer	—	Quantity Ordered	
total_amount	decimal	10,2	Total Price (Qty × Unit Price)	
shipping_address	string	100	Destination Address	
order_status	string	15	placed / dispatched / return in progress / returned	
order_date	date	—	Date of Order	

7. Payment

Field Name	Data Type	Size	Description	Key
paymentID	string	10	Payment ID	PK
invoiceID	string	10	Invoice ID	FK → Invoice.invoiceID
method	string	20	Payment Method	
status	string	15	Payment Status	
payment_date	date	—	Date of Payment	

8. Invoice

Field Name	Data Type	Size	Description	Key
invoiceID	string	10	Invoice ID	PK
orderID	string	10	Order ID	FK → Orders.orderID
buyerID	string	10	Buyer ID	FK → Buyer.BID
amt	decimal	10,2	Total Amount	
tax_amt	decimal	10,2	Tax Amount	

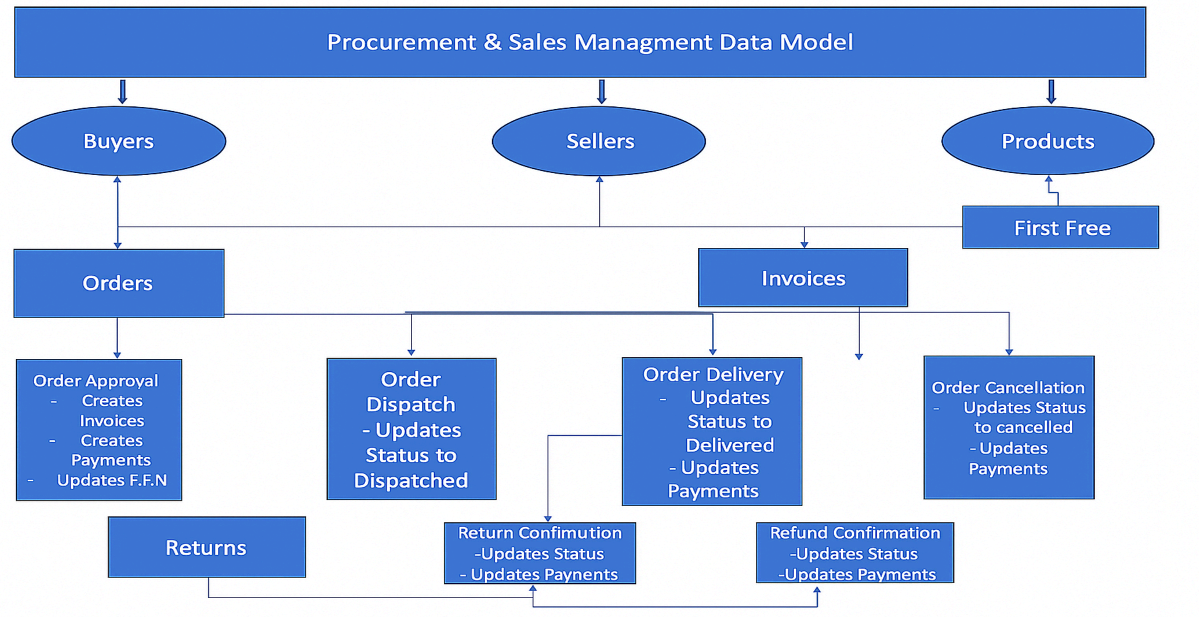
9. FirstFreeNumber

Field Name	Data Type	Size	Description	Key
orig	string	10	Module Origin	PK
ival	int	—	Initial Value	
ffnu	int	—	Next Free Number	

10. Return

Field Name	Data Type	Size	Description	Key
retID	string	10	Return ID	PK
orderID	string	10	Order ID	FK → Orders.orderID
status	string	20	Return Status (in progress, returned)	
refund_status	string	20	Refunded / Not Refunded	
refund_date	date	—	Date of Refund	
pickup_date	date	—	Date of Pickup	

ER diagram:


Procurement & Sales Managment Data Model
Buyers
Sellers
Products
First Free
Orders
Invoices
Order Approval - Creates Invoices - Creates Payments - Updates F.F.N
Order Dispatch - Updates Status to Dispatched
Order Delivery - Updates Status to Delivered - Updates Payments
Order Cancellation - Updates Status to cancelled - Updates Payments
Returns
Return Confimution -Updates Status - Updates Paynents
Refund Confirmation -Updates Status -Updates Payments

Screenshots & Report

This module contains all the Screenshots and reports (i.e Table definitions-Indices & References, Maintain session -overview session & Details sessions, print sessions) generated and their related data which are generated during the process of development

6.1 Tables Definitions

Countries table:

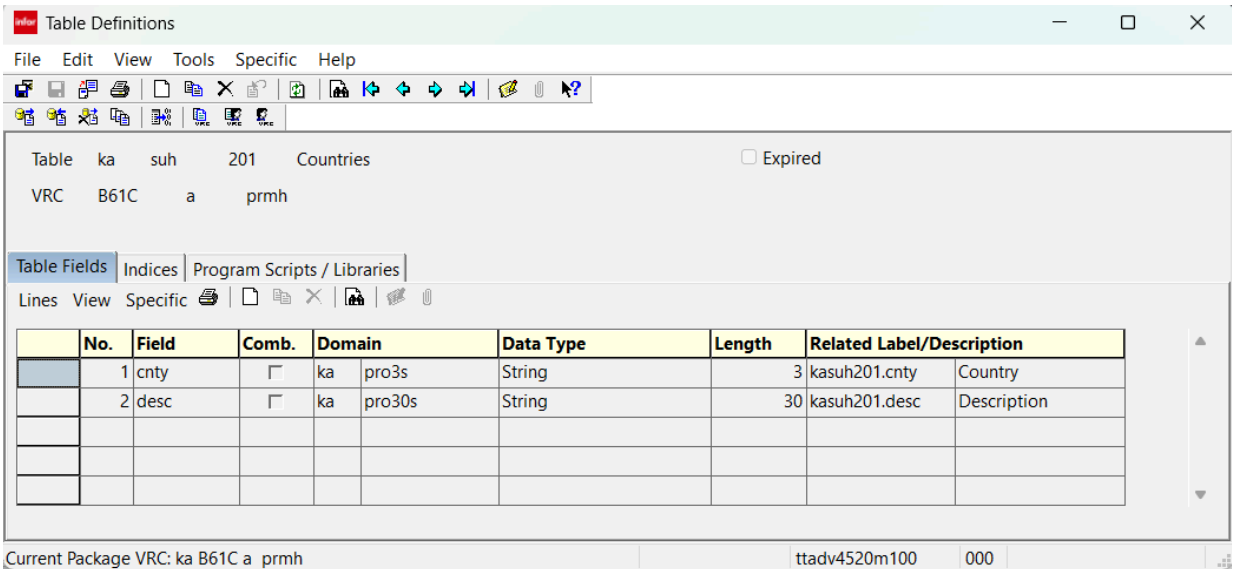


Table Definitions
File Edit View Tools Specific Help
Table Fields
Indices
Program Scripts / Libraries
Lines View Specific

Table ka suh 201 Countries □ Expired

VRC B61C a prmh

	No.	Field	Comb.	Domain		Data Type	Length	Related Label/Description	
	1	cnty	□	ka	pro3s	String	3	kasuh201.cnty	Country
	2	desc	□	ka	pro30s	String	30	kasuh201.desc	Description

Current Package VRC: ka B61C a prmh ttadv4520m100 000

Indices:

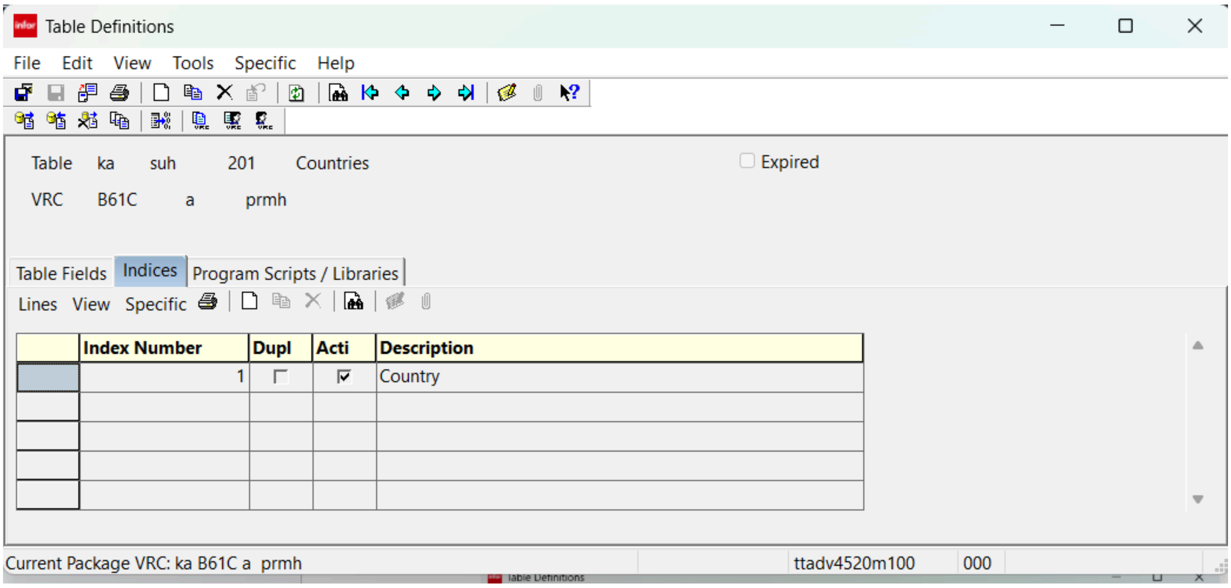


Table Definitions
File Edit View Tools Specific Help
Table Fields
Indices
Program Scripts / Libraries
Lines View Specific

Table ka suh 201 Countries □ Expired

VRC B61C a prmh

	Index Number	Dupl	Acti	Description
	1	□	☑	Country

Current Package VRC: ka B61C a prmh ttadv4520m100 000

Buyers:

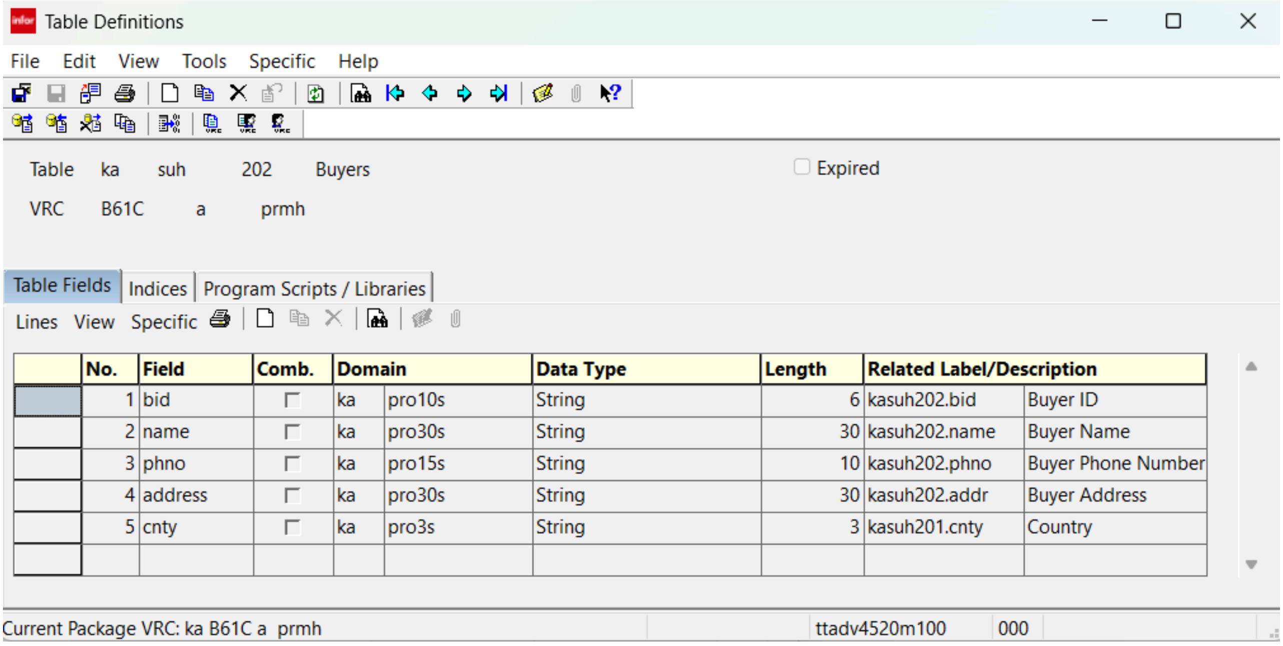


Table Definitions
File Edit View Tools Specific Help
Table Fields
Indices
Program Scripts / Libraries
Lines View Specific

Table ka suh 202 Buyers □ Expired

VRC B61C a prmh

	No.	Field	Comb.	Domain		Data Type	Length	Related Label/Description	
	1	bid	□	ka	pro10s	String	6	kasuh202.bid	Buyer ID
	2	name	□	ka	pro30s	String	30	kasuh202.name	Buyer Name
	3	phno	□	ka	pro15s	String	10	kasuh202.phno	Buyer Phone Number
	4	address	□	ka	pro30s	String	30	kasuh202.addr	Buyer Address
	5	cnty	□	ka	pro3s	String	3	kasuh201.cnty	Country

Current Package VRC: ka B61C a prmh ttadv4520m100 000

Indices:

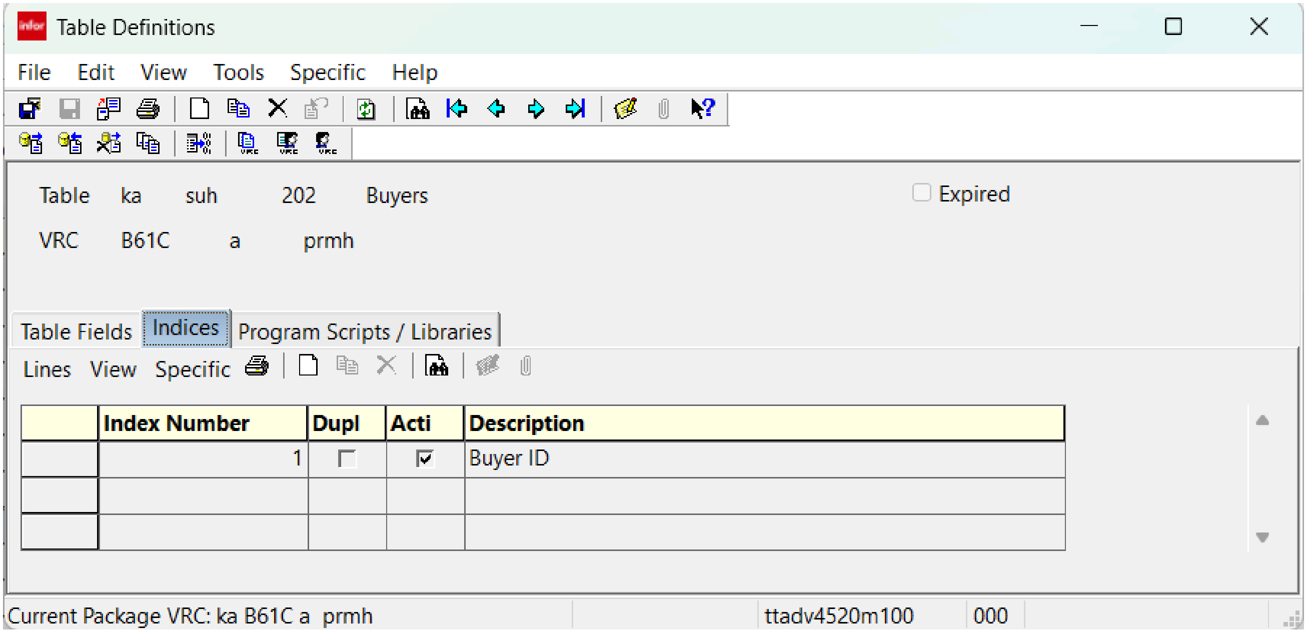


Table Definitions
File Edit View Tools Specific Help
Table Fields
Indices
Program Scripts / Libraries
Lines View Specific

Table ka suh 202 Buyers □ Expired

VRC B61C a prmh

	Index Number	Dupl	Acti	Description
	1	□	☑	Buyer ID

Current Package VRC: ka B61C a prmh ttadv4520m100 000

Seller:

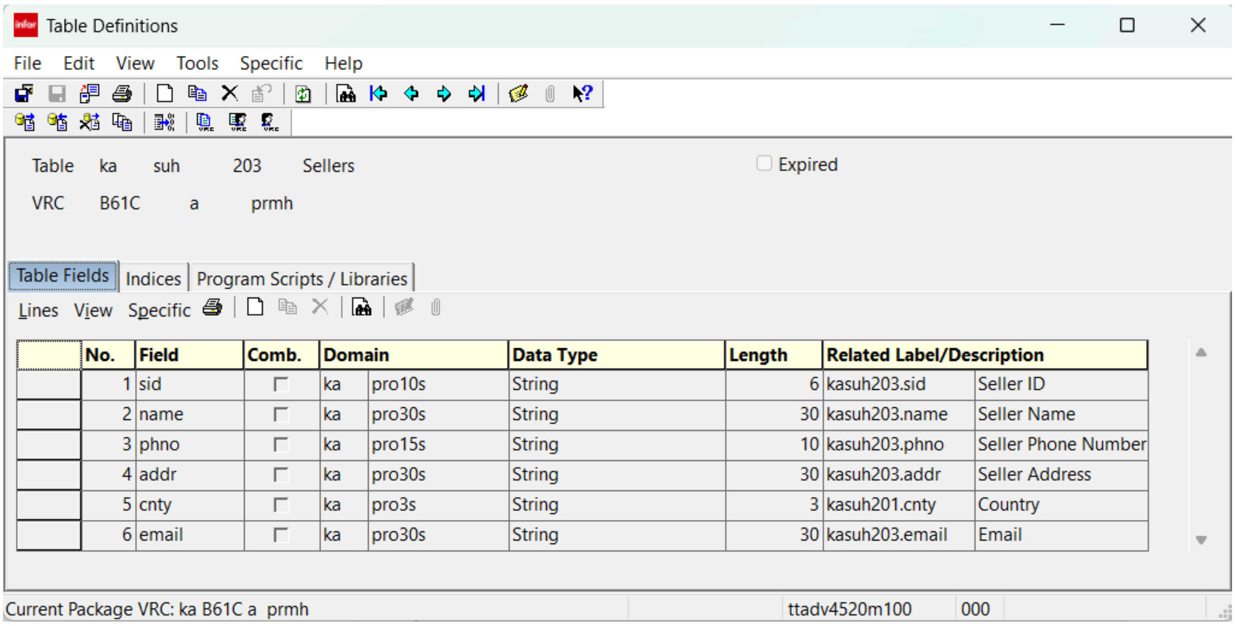


Table Definitions
File Edit View Tools Specific Help
Table Fields Indices Program Scripts / Libraries
Lines View Specific

Table ka suh 203 Sellers □ Expired

VRC B61C a prmh

	No.	Field	Comb.	Domain		Data Type	Length	Related Label/Description	
	1	sid	□	ka	pro10s	String	6	kasuh203.sid	Seller ID
	2	name	□	ka	pro30s	String	30	kasuh203.name	Seller Name
	3	phno	□	ka	pro15s	String	10	kasuh203.phno	Seller Phone Number
	4	addr	□	ka	pro30s	String	30	kasuh203.addr	Seller Address
	5	cnty	□	ka	pro3s	String	3	kasuh201.cnty	Country
	6	email	□	ka	pro30s	String	30	kasuh203.email	Email

Current Package VRC: ka B61C a prmh ttadv4520m100 000

Indices :

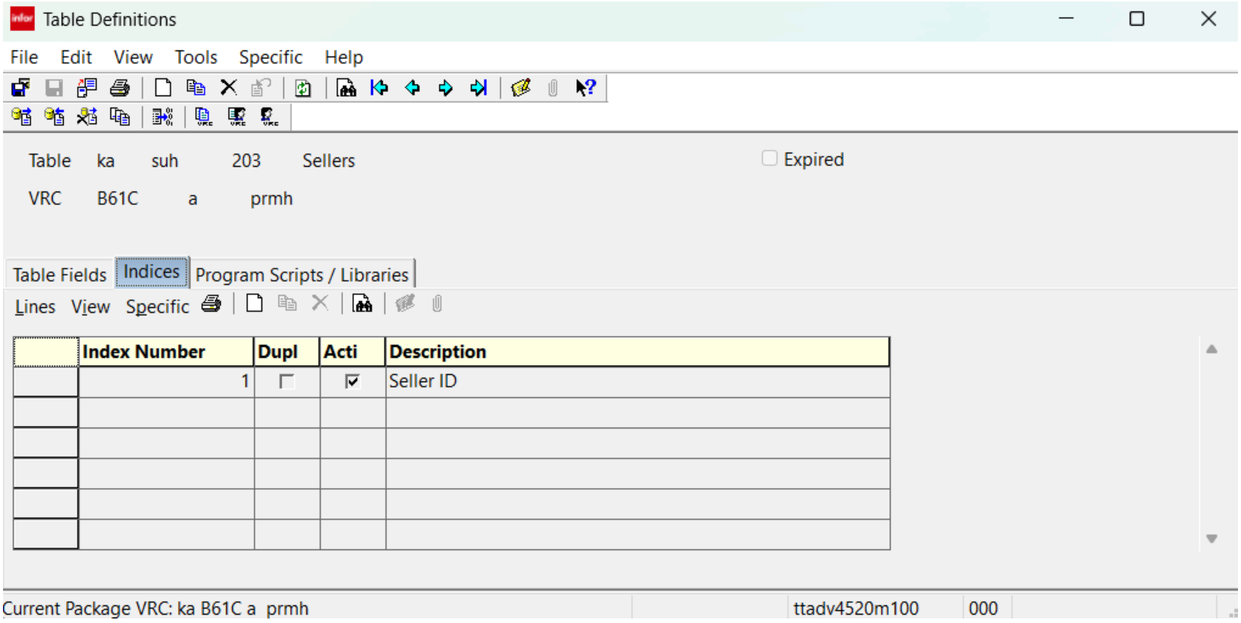


Table Definitions
File Edit View Tools Specific Help
Table Fields Indices Program Scripts / Libraries
Lines View Specific

Table ka suh 203 Sellers □ Expired

VRC B61C a prmh

	Index Number	Dupl	Acti	Description
	1	□	☑	Seller ID

Current Package VRC: ka B61C a prmh ttadv4520m100 000

Product Group:

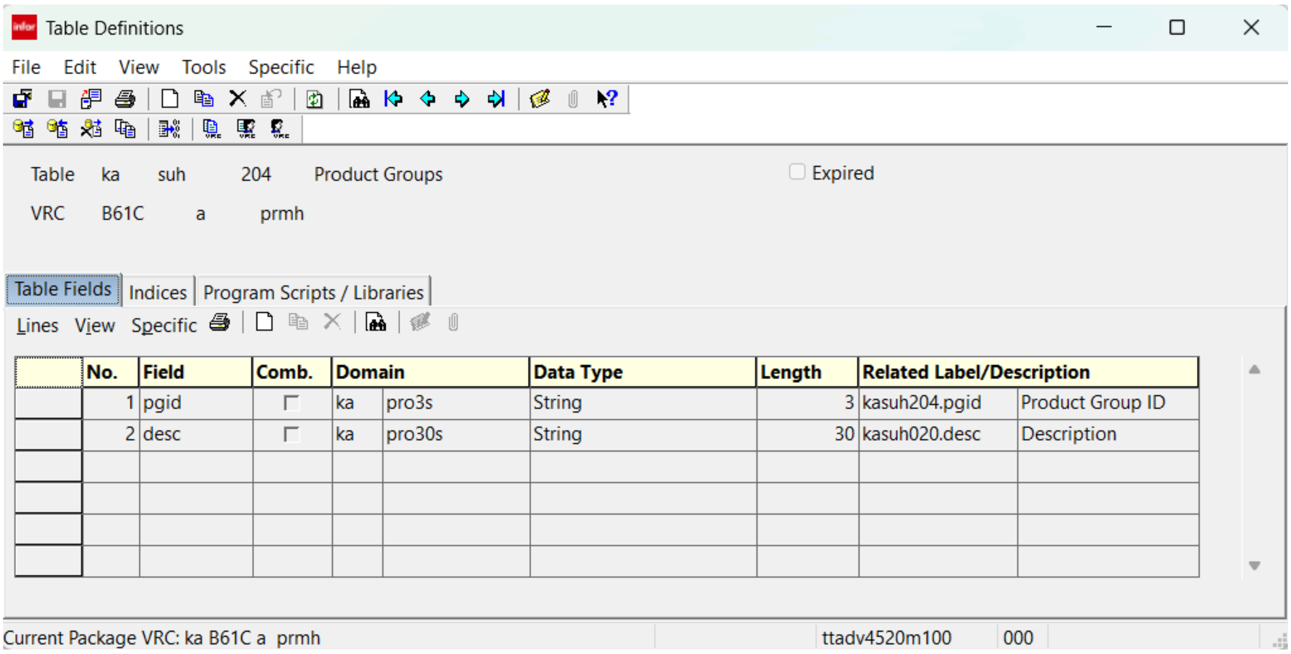


Table Definitions
File Edit View Tools Specific Help
Table Fields
Indices
Program Scripts / Libraries
Lines View Specific

Table ka suh 204 Product Groups □ Expired

VRC B61C a prmh

No.	Field	Comb.	Domain		Data Type	Length	Related Label/Description	
1	pgid	□	ka	pro3s	String	3	kasuh204.pgid	Product Group ID
2	desc	□	ka	pro30s	String	30	kasuh020.desc	Description

Current Package VRC: ka B61C a prmh ttadv4520m100 000

Indices:

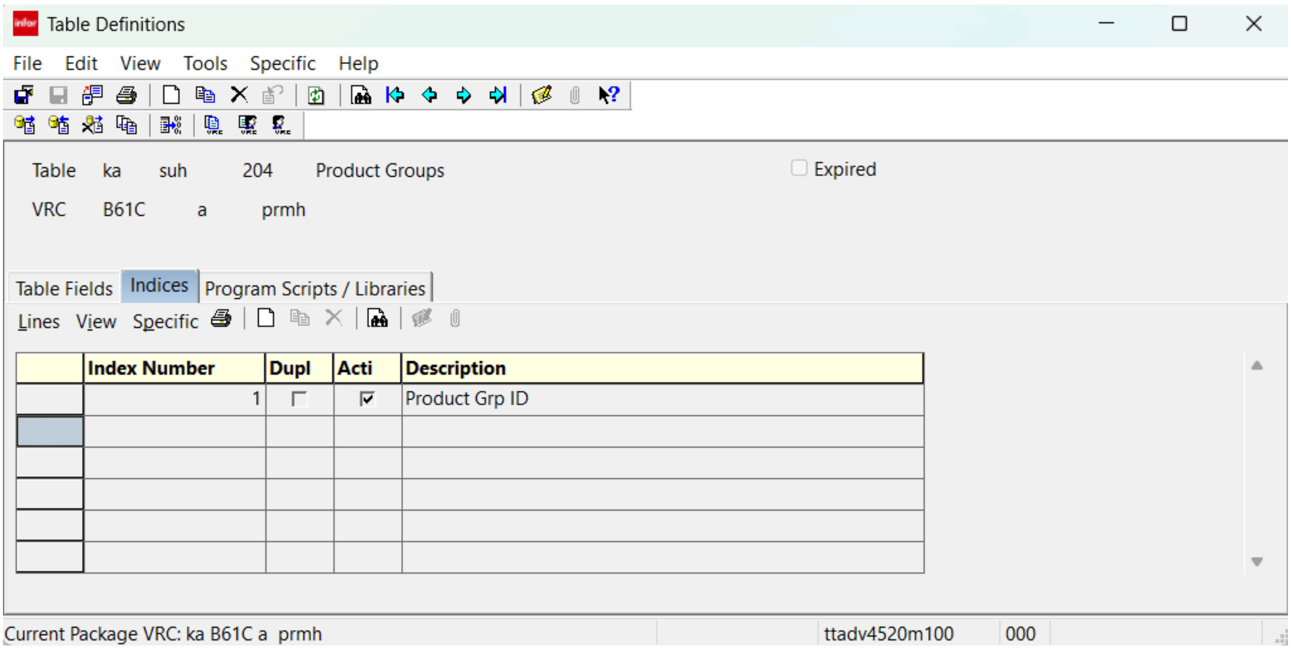


Table Definitions
File Edit View Tools Specific Help
Table Fields
Indices
Program Scripts / Libraries
Lines View Specific

Table ka suh 204 Product Groups □ Expired

VRC B61C a prmh

Index Number	Dupl	Acti	Description
1	□	☑	Product Grp ID

Current Package VRC: ka B61C a prmh ttadv4520m100 000

Products:

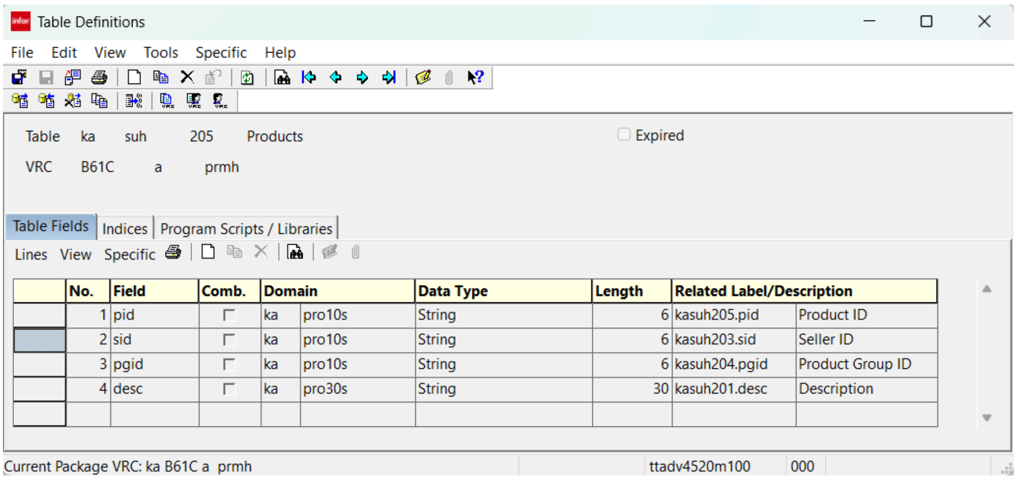


Table Definitions
File Edit View Tools Specific Help
Table Fields
Indices
Program Scripts / Libraries
Lines View Specific

Table ka suh 205 Products ☐ Expired

VRC B61C a prmh

No.	Field	Comb.	Domain		Data Type	Length	Related Label/Description	
1	pid	☐	ka	pro10s	String	6	kasuh205.pid	Product ID
2	sid	☐	ka	pro10s	String	6	kasuh203.sid	Seller ID
3	pgid	☐	ka	pro10s	String	6	kasuh204.pgid	Product Group ID
4	desc	☐	ka	pro30s	String	30	kasuh201.desc	Description

Current Package VRC: ka B61C a prmh ttadv4520m100 000

Indices:

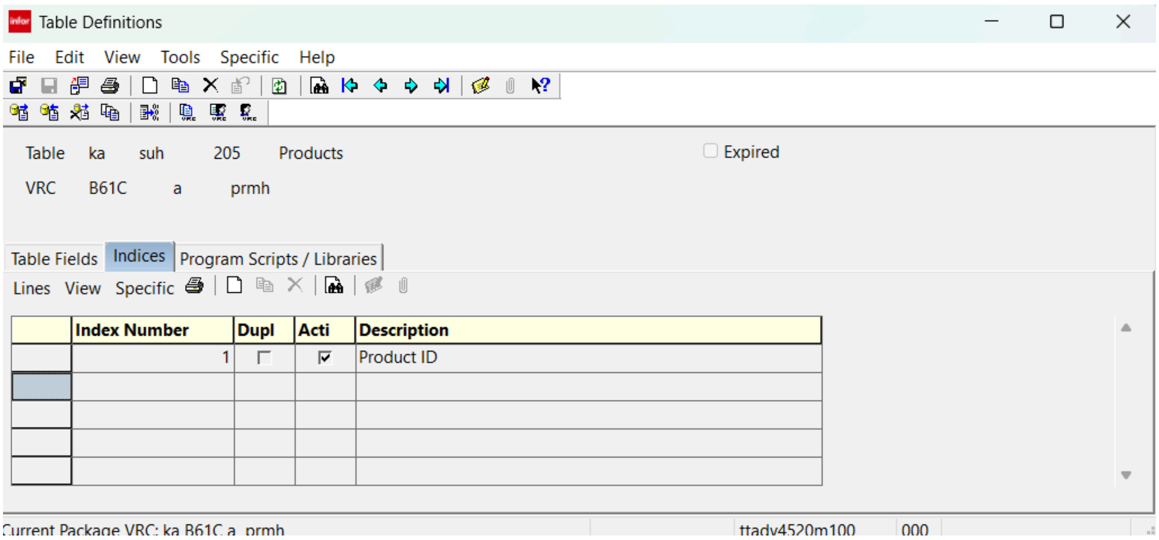


Table Definitions
File Edit View Tools Specific Help
Table Fields
Indices
Program Scripts / Libraries
Lines View Specific

Table ka suh 205 Products ☐ Expired

VRC B61C a prmh

Index Number	Dupl	Acti	Description
1	☐	☑	Product ID

Current Package VRC: ka B61C a prmh ttadv4520m100 000

Reference:

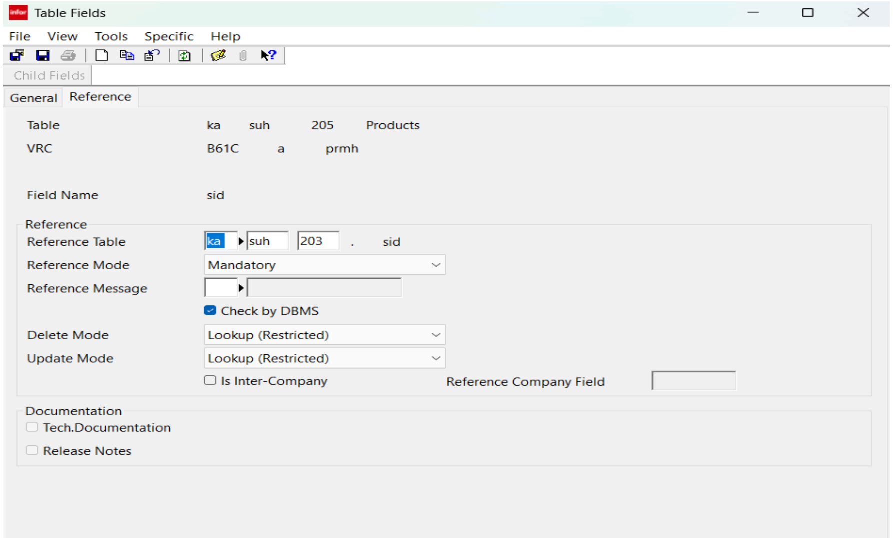


Table ka suh 205 Products

VRC B61C a prmh

Field Name sid

Reference

Reference Table ka suh 203 . sid

Reference Mode Mandatory

Reference Message

☑ Check by DBMS

Delete Mode Lookup (Restricted)

Update Mode Lookup (Restricted)

☐ Is Inter-Company Reference Company Field

Documentation

☐ Tech.Documentation

☐ Release Notes

Orders:

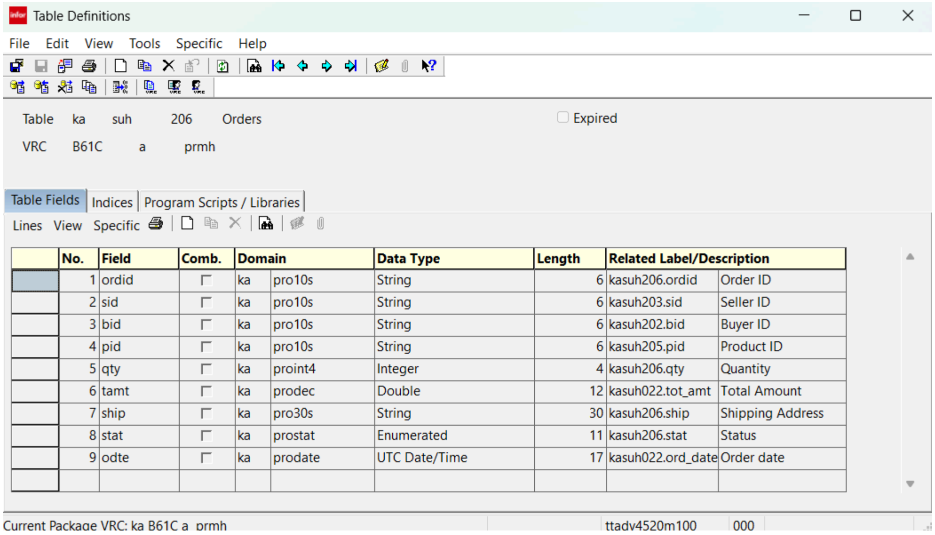


Table Definitions
File Edit View Tools Specific Help
Table Fields
Indices
Program Scripts / Libraries
Lines View Specific

Table ka suh 206 Orders □ Expired

VRC B61C a prmh

No.	Field	Comb.	Domain		Data Type	Length	Related Label/Description	
1	ordid	□	ka	pro10s	String	6	kasuh206.ordid	Order ID
2	sid	□	ka	pro10s	String	6	kasuh203.sid	Seller ID
3	bid	□	ka	pro10s	String	6	kasuh202.bid	Buyer ID
4	pid	□	ka	pro10s	String	6	kasuh205.pid	Product ID
5	qty	□	ka	proint4	Integer	4	kasuh206.qty	Quantity
6	tamt	□	ka	prodec	Double	12	kasuh022.tot_amt	Total Amount
7	ship	□	ka	pro30s	String	30	kasuh206.ship	Shipping Address
8	stat	□	ka	prostat	Enumerated	11	kasuh206.stat	Status
9	odte	□	ka	prodate	UTC Date/Time	17	kasuh022.ord_date	Order date

Current Package VRC: ka B61C a prmh ttadv4520m100 000

Indices:

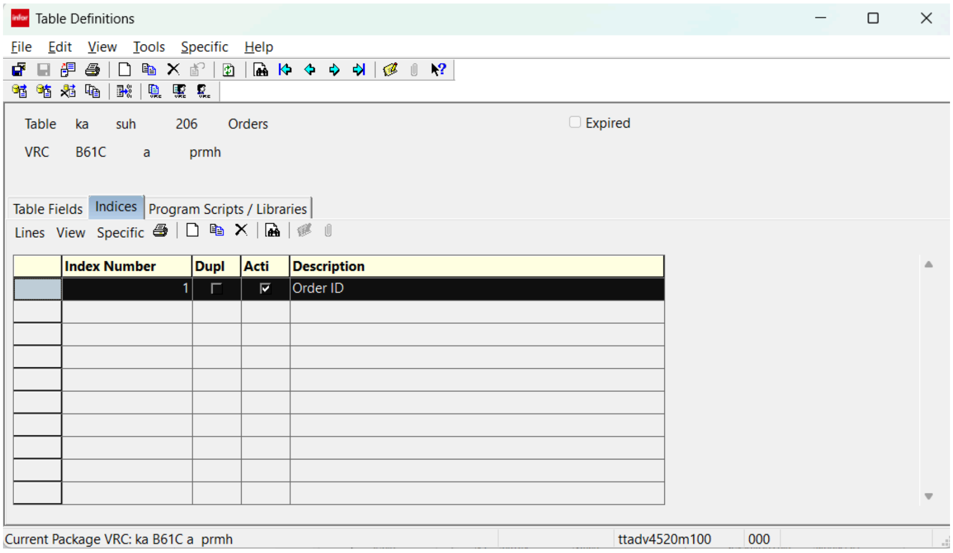


Table Definitions
File Edit View Tools Specific Help
Table Fields
Indices
Program Scripts / Libraries
Lines View Specific

Table ka suh 206 Orders □ Expired

VRC B61C a prmh

Index Number	Dupl	Acti	Description
1	□	☑	Order ID

Current Package VRC: ka B61C a prmh ttadv4520m100 000

References for Order Table:

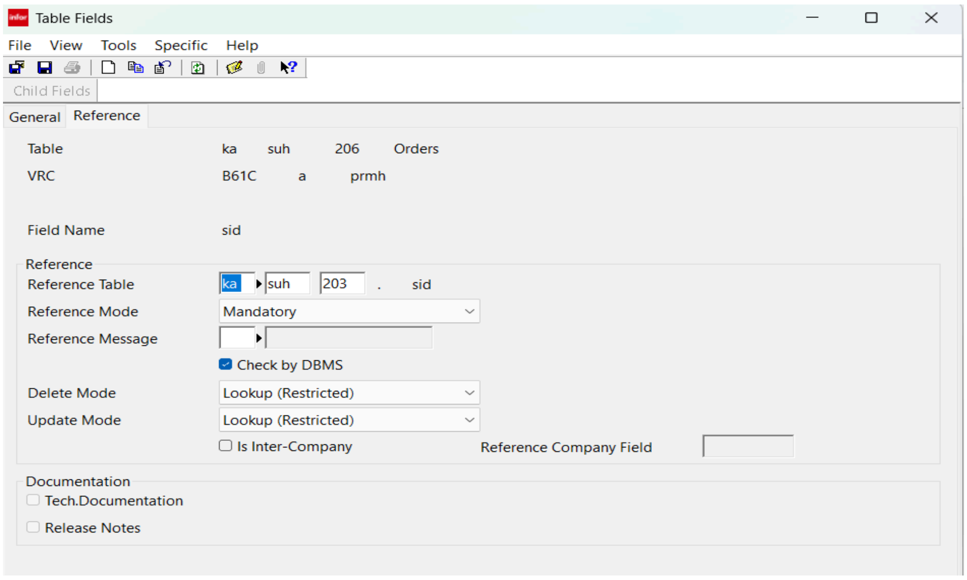


Table ka suh 206 Orders

VRC B61C a prmh

Field Name sid

Reference

Reference Table ka suh 203 . sid

Reference Mode Mandatory

Reference Message

☑ Check by DBMS

Delete Mode Lookup (Restricted)

Update Mode Lookup (Restricted)

□ Is Inter-Company Reference Company Field

Documentation

□ Tech.Documentation

□ Release Notes

Indices:

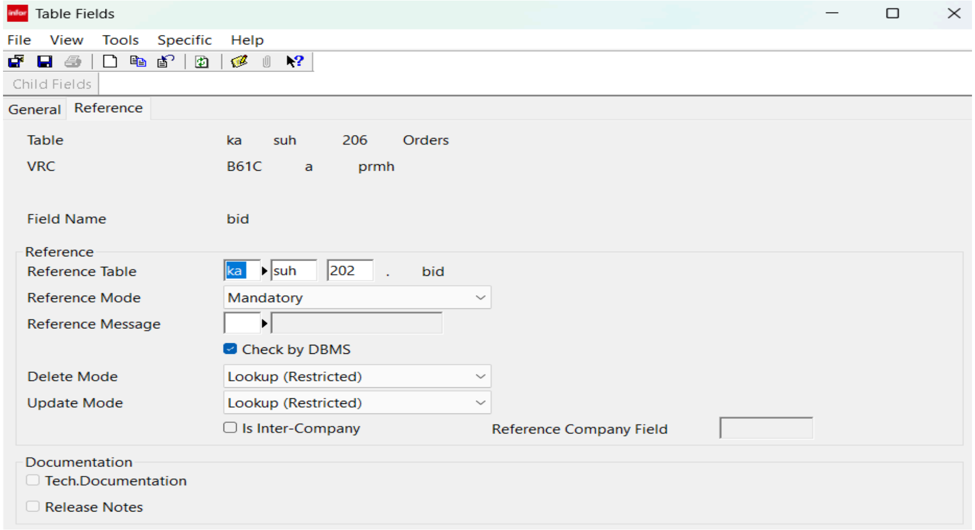


Table ka suh 206 Orders

VRC B61C a prmh

Field Name bid

Reference

Reference Table ka suh 202 . bid

Reference Mode Mandatory

Reference Message

☑ Check by DBMS

Delete Mode Lookup (Restricted)

Update Mode Lookup (Restricted)

□ Is Inter-Company Reference Company Field

Documentation

□ Tech.Documentation

□ Release Notes

References:

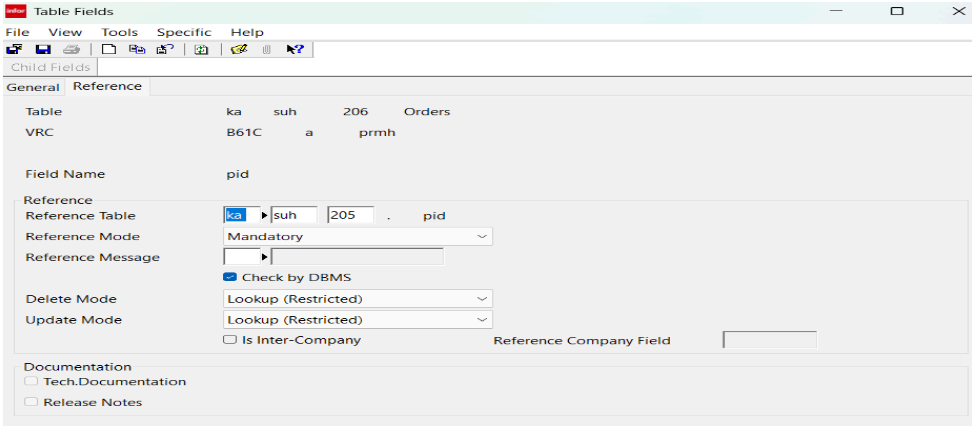


Table ka suh 206 Orders

VRC B61C a prmh

Field Name pid

Reference

Reference Table ka suh 205 . pid

Reference Mode Mandatory

Reference Message

☑ Check by DBMS

Delete Mode Lookup (Restricted)

Update Mode Lookup (Restricted)

□ Is Inter-Company Reference Company Field

Documentation

□ Tech.Documentation

□ Release Notes

Invoice:

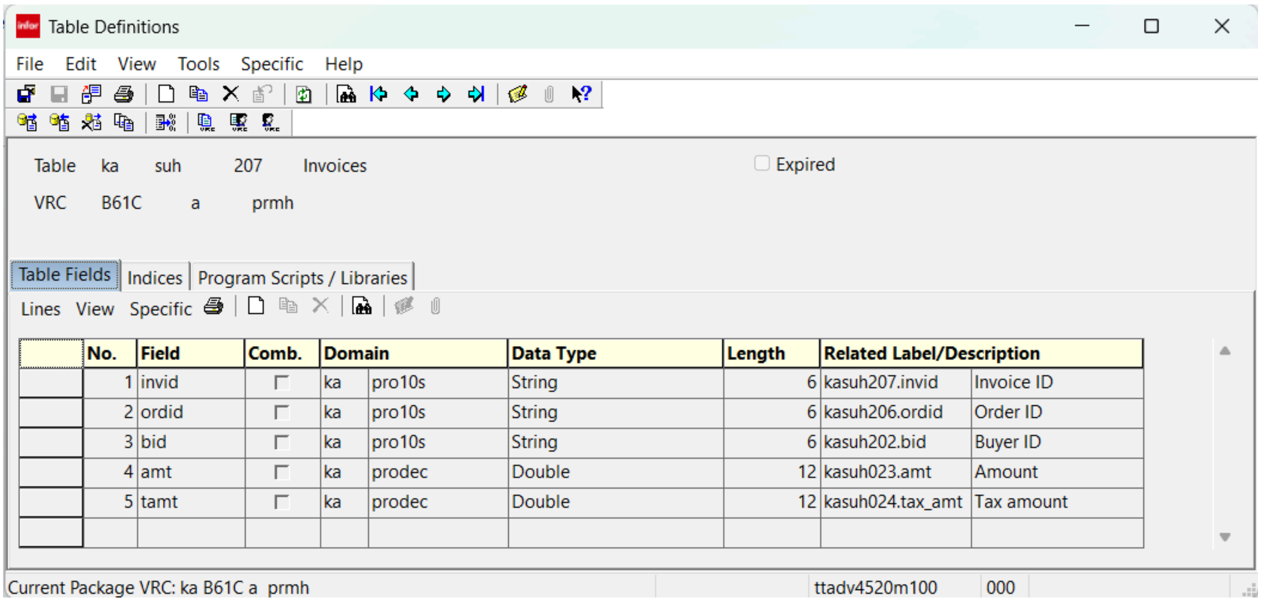


Table Definitions
File Edit View Tools Specific Help
Table Fields Indices Program Scripts / Libraries
Lines View Specific

Table ka suh 207 Invoices □ Expired

VRC B61C a prmh

	No.	Field	Comb.	Domain		Data Type	Length	Related Label/Description	
	1	invid	□	ka	pro10s	String	6	kasuh207.invid	Invoice ID
	2	ordid	□	ka	pro10s	String	6	kasuh206.ordid	Order ID
	3	bid	□	ka	pro10s	String	6	kasuh202.bid	Buyer ID
	4	amt	□	ka	prodec	Double	12	kasuh023.amt	Amount
	5	tamt	□	ka	prodec	Double	12	kasuh024.tax_amt	Tax amount

Current Package VRC: ka B61C a prmh ttadv4520m100 000

Indices:

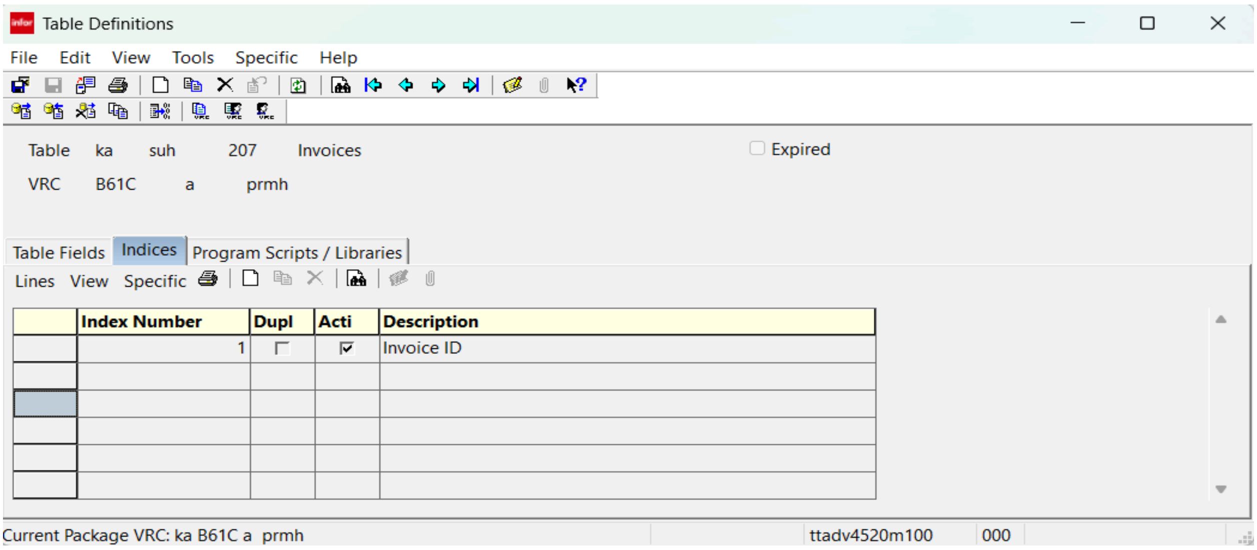


Table Definitions
File Edit View Tools Specific Help
Table Fields
Indices
Program Scripts / Libraries
Lines View Specific

Table ka suh 207 Invoices

□ Expired

VRC B61C a prmh

	Index Number	Dupl	Acti	Description
	1	□	☑	Invoice ID

Current Package VRC: ka B61C a prmh ttadv4520m100 000

References:

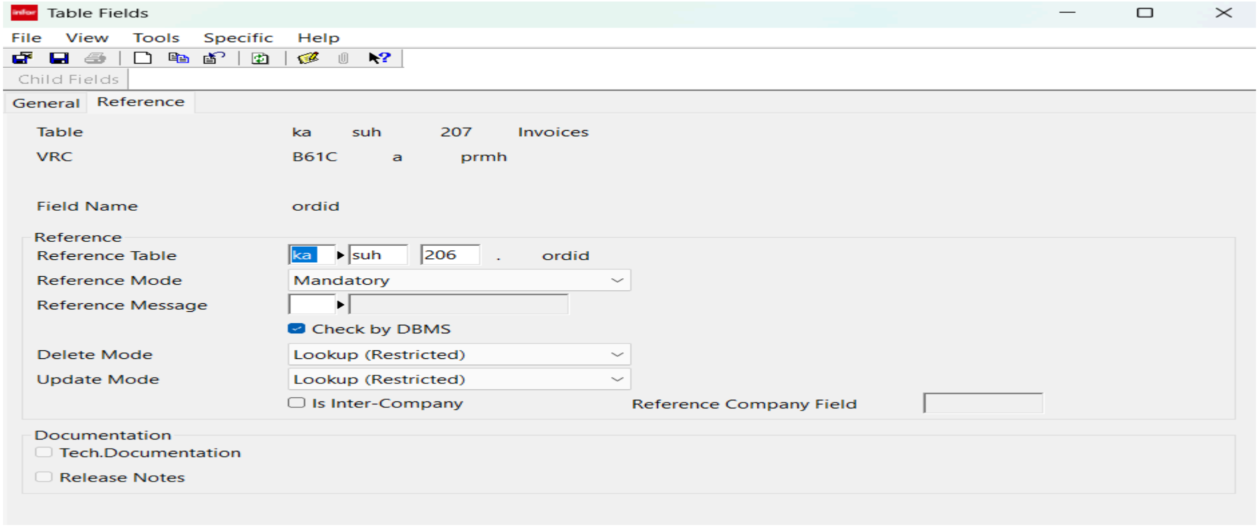


Table ka suh 207 Invoices

VRC B61C a prmh

Field Name ordid

Reference

Reference Table ka suh 206 . ordid

Reference Mode Mandatory

Reference Message

☑ Check by DBMS

Delete Mode Lookup (Restricted)

Update Mode Lookup (Restricted)

□ Is Inter-Company Reference Company Field

Documentation

□ Tech.Documentation

□ Release Notes

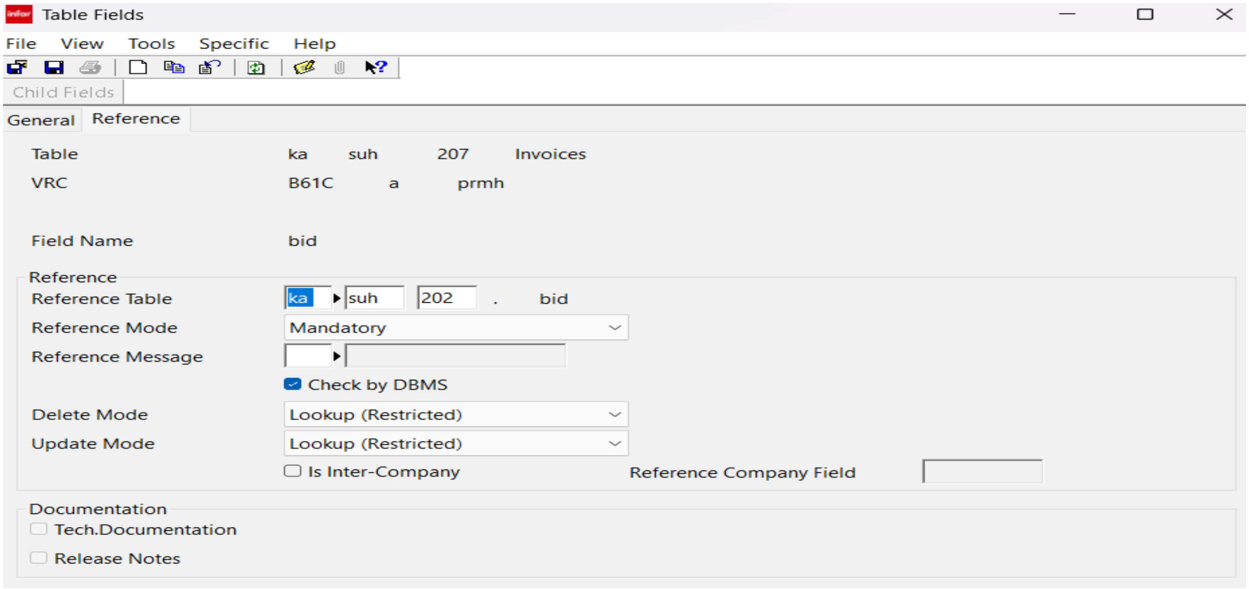


Table ka suh 207 Invoices

VRC B61C a prmh

Field Name bid

Reference

Reference Table ka suh 202 . bid

Reference Mode Mandatory

Reference Message

☑ Check by DBMS

Delete Mode Lookup (Restricted)

Update Mode Lookup (Restricted)

□ Is Inter-Company Reference Company Field

Documentation

□ Tech.Documentation

□ Release Notes

Payments:

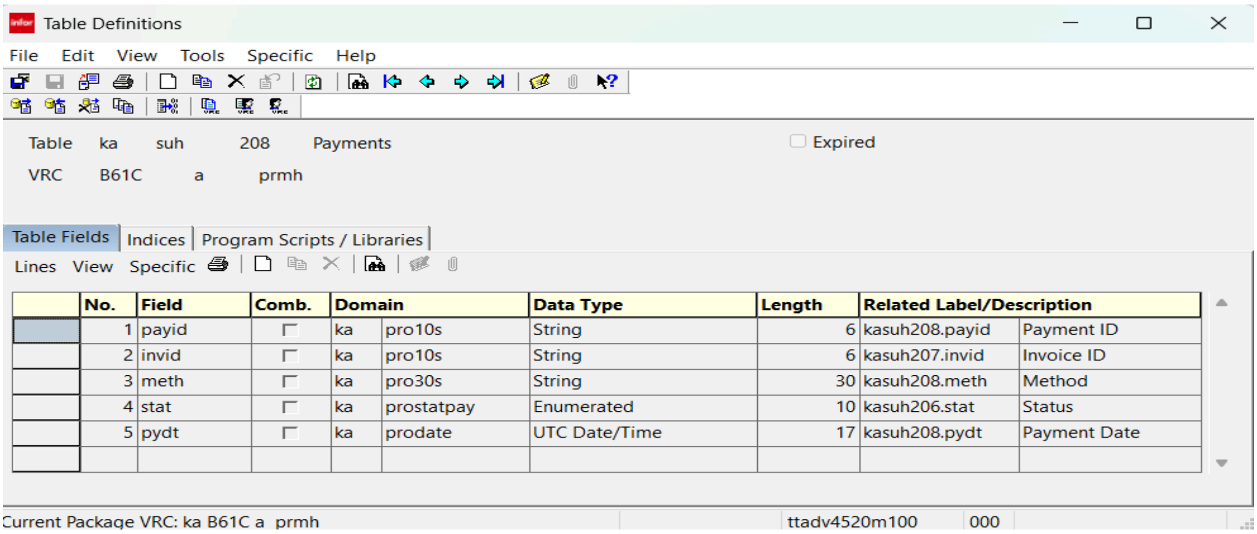


Table Definitions
File Edit View Tools Specific Help
Table Fields
Indices
Program Scripts / Libraries
Lines View Specific

Table	ka	suh	208	Payments
VRC	B61C	a	prmh	

□ Expired

No.	Field	Comb.	Domain		Data Type	Length	Related Label/Description	
1	payid	□	ka	pro10s	String	6	kasuh208.payid	Payment ID
2	invid	□	ka	pro10s	String	6	kasuh207.invid	Invoice ID
3	meth	□	ka	pro30s	String	30	kasuh208.meth	Method
4	stat	□	ka	prostatpay	Enumerated	10	kasuh206.stat	Status
5	pydt	□	ka	prodate	UTC Date/Time	17	kasuh208.pydt	Payment Date

Current Package VRC: ka B61C a prmh ttadv4520m100 000

Indices:

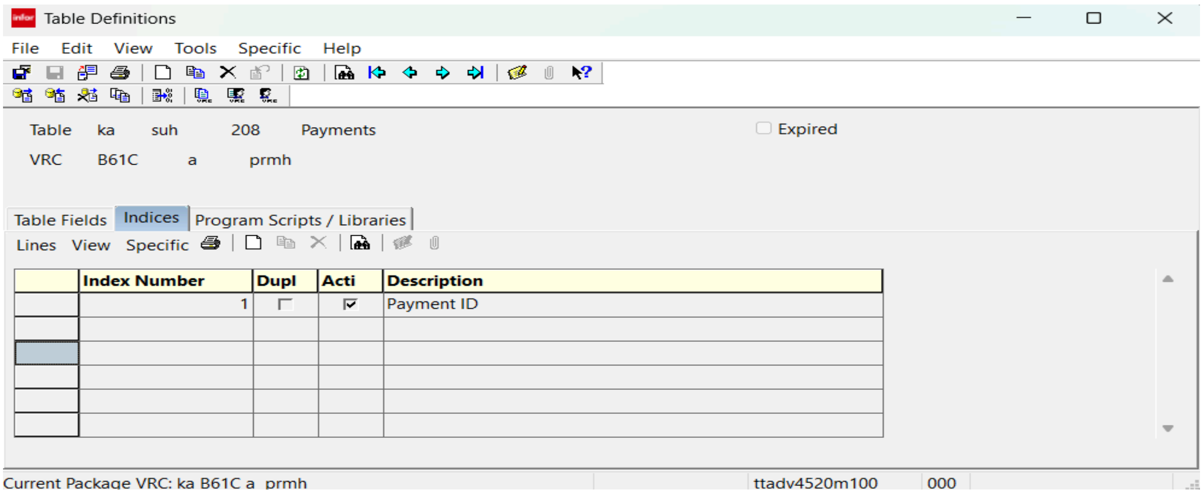


Table Definitions
File Edit View Tools Specific Help
Table Fields
Indices
Program Scripts / Libraries
Lines View Specific

Table	ka	suh	208	Payments
VRC	B61C	a	prmh	

□ Expired

Index Number	Dupl	Acti	Description
1	□	☑	Payment ID

Current Package VRC: ka B61C a prmh ttadv4520m100 000

References:

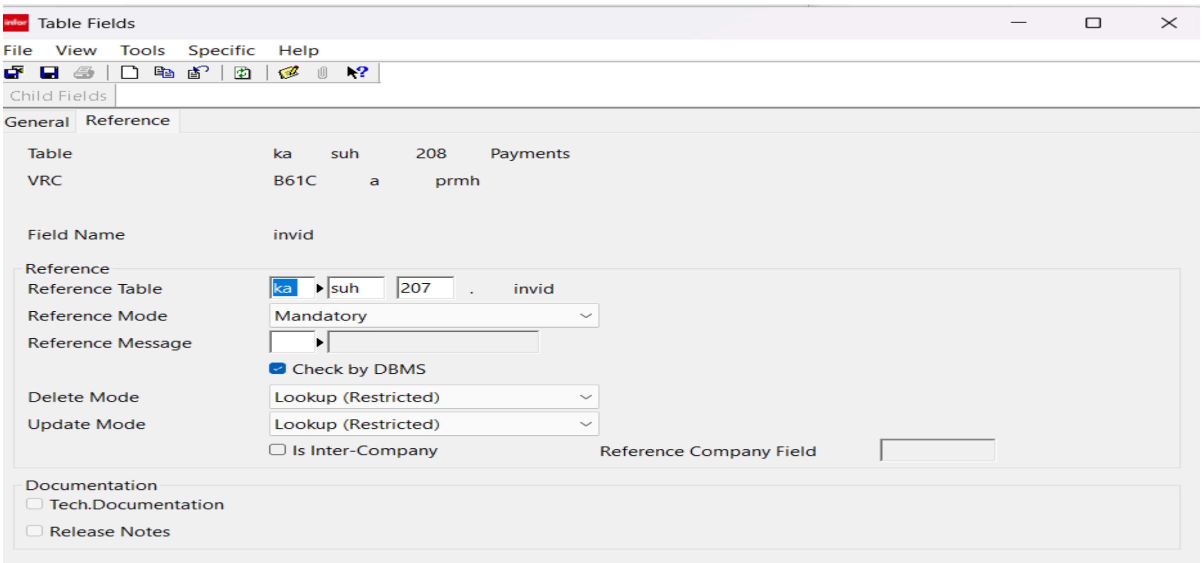


Table	ka	suh	208	Payments
VRC	B61C	a	prmh	

Field Name invid

Reference

Reference Table ka suh 207 . invid

Reference Mode Mandatory

Reference Message

☑ Check by DBMS

Delete Mode Lookup (Restricted)

Update Mode Lookup (Restricted)

□ Is Inter-Company Reference Company Field

Documentation

□ Tech.Documentation

□ Release Notes

Returns:

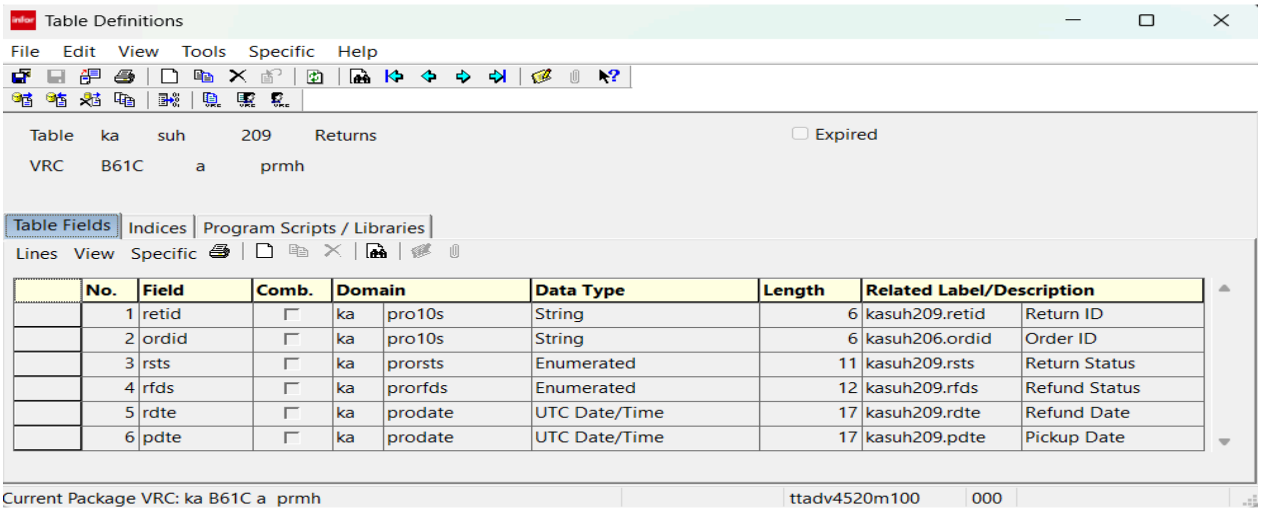


Table Definitions
File Edit View Tools Specific Help
Table Fields Indices Program Scripts / Libraries
Lines View Specific

Table ka suh 209 Returns □ Expired

VRC B61C a prmh

No.	Field	Comb.	Domain		Data Type	Length	Related Label/Description	
1	retid	□	ka	pro10s	String	6	kasuh209.retid	Return ID
2	ordid	□	ka	pro10s	String	6	kasuh206.ordid	Order ID
3	rsts	□	ka	prorsts	Enumerated	11	kasuh209.rsts	Return Status
4	rfds	□	ka	prorfds	Enumerated	12	kasuh209.rfds	Refund Status
5	rdte	□	ka	prodate	UTC Date/Time	17	kasuh209.rdte	Refund Date
6	pdte	□	ka	prodate	UTC Date/Time	17	kasuh209.pdte	Pickup Date

Current Package VRC: ka B61C a prmh ttadv4520m100 000

Indices:

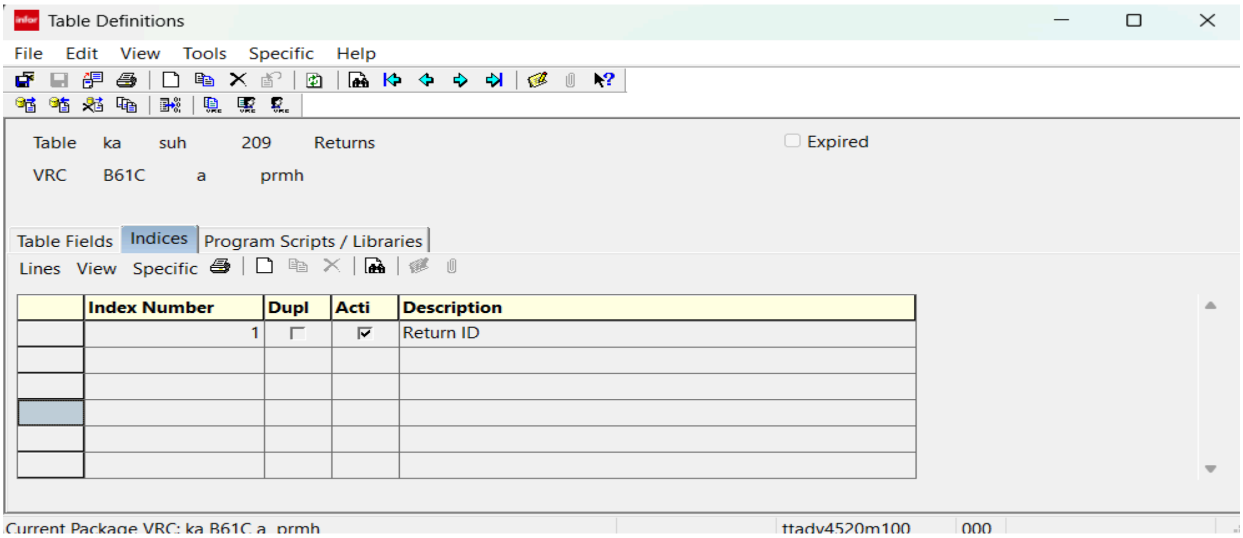


Table Definitions
File Edit View Tools Specific Help
Table Fields Indices Program Scripts / Libraries
Lines View Specific

Table ka suh 209 Returns □ Expired

VRC B61C a prmh

Index Number	Dupl	Acti	Description
1	□	☑	Return ID

Current Package VRC: ka B61C a prmh ttadv4520m100 000

References:

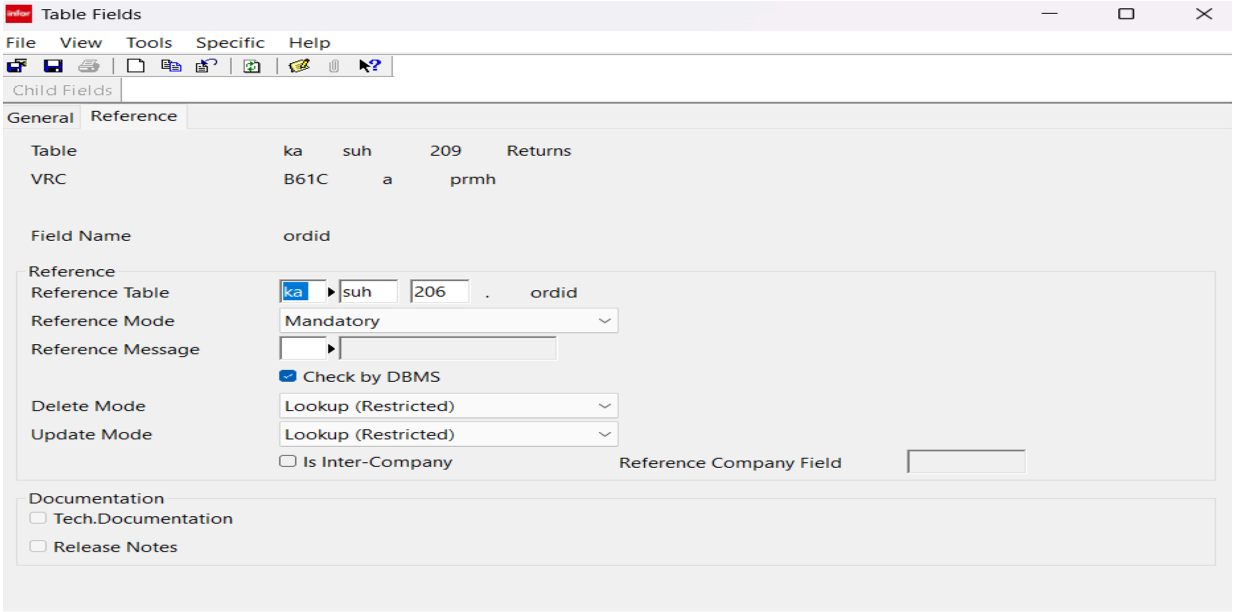


Table	ka suh 209 Returns
VRC	B61C a prmh
Field Name	ordid

Reference

Reference Table	ka suh 206 . ordid
Reference Mode	Mandatory
Reference Message	
	☑ Check by DBMS
Delete Mode	Lookup (Restricted)
Update Mode	Lookup (Restricted)
	□ Is Inter-Company Reference Company Field

Documentation

□ Tech.Documentation

□ Release Notes

First Free Number table:

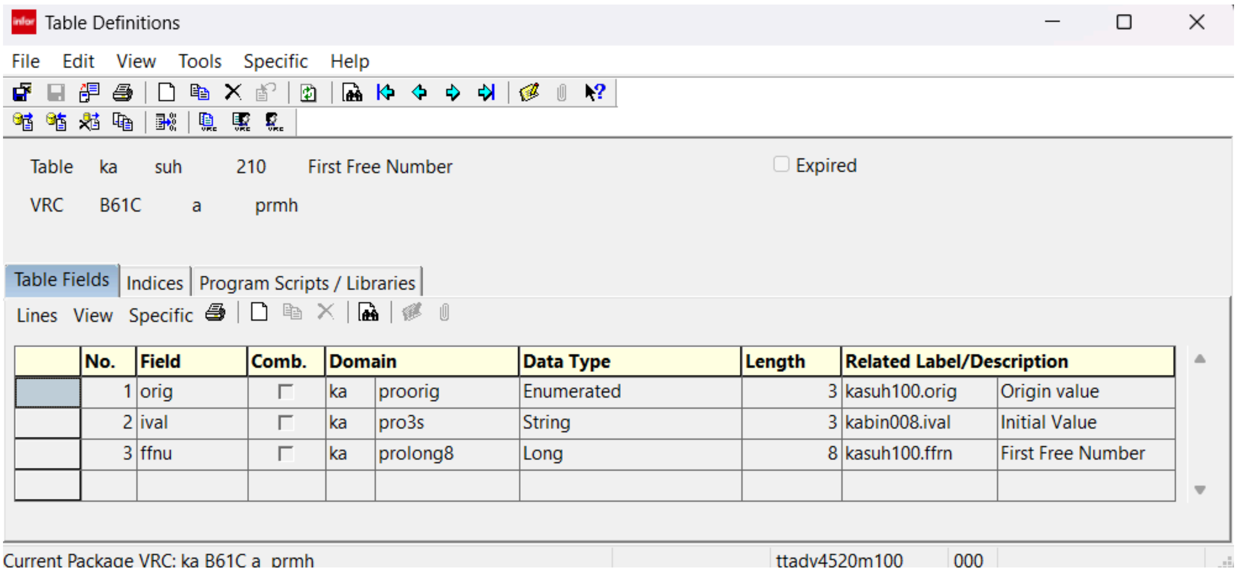


Table Definitions
File Edit View Tools Specific Help
Table Fields Indices Program Scripts / Libraries
Lines View Specific

Table ka suh 210 First Free Number □ Expired

VRC B61C a prmh

	No.	Field	Comb.	Domain		Data Type	Length	Related Label/Description	
	1	orig	□	ka	proorig	Enumerated	3	kasuh100.orig	Origin value
	2	ival	□	ka	pro3s	String	3	kabin008.ival	Initial Value
	3	ffnu	□	ka	prolong8	Long	8	kasuh100.ffrn	First Free Number

Current Package VRC: ka B61C a prmh ttadv4520m100 000

Indices:

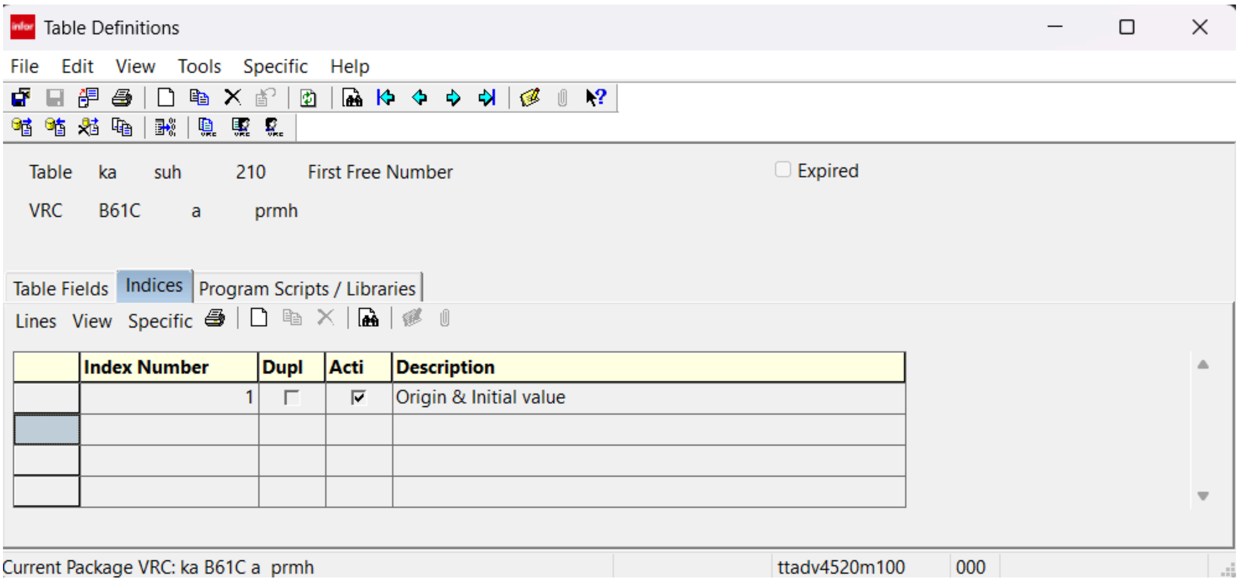


Table Definitions
File Edit View Tools Specific Help
Table Fields Indices Program Scripts / Libraries
Lines View Specific

Table ka suh 210 First Free Number □ Expired

VRC B61C a prmh

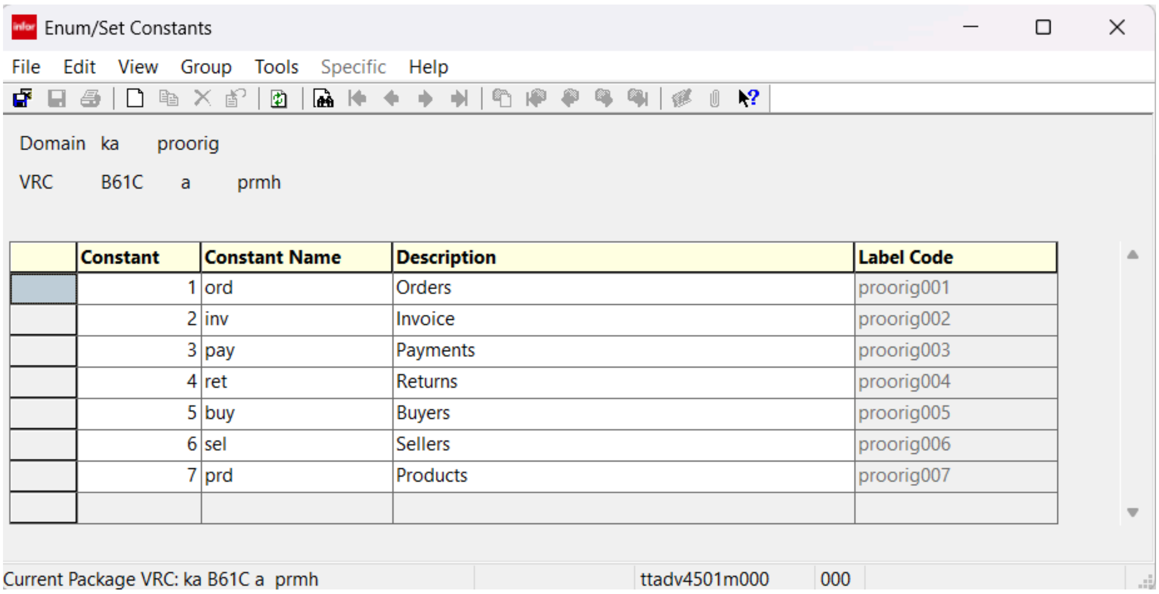
	Index Number	Dupl	Acti	Description
	1	□	☑	Origin & Initial value

Current Package VRC: ka B61C a prmh ttadv4520m100 000

6.2 Enum Values for the Enumerated Domains:

Enumerated domains in Infor LN restrict field values using listboxes or optionsets.
Listboxes display a predefined list of selectable enum values.
Optionsets show only specified options, hiding all others for controlled data entry.

Origin:



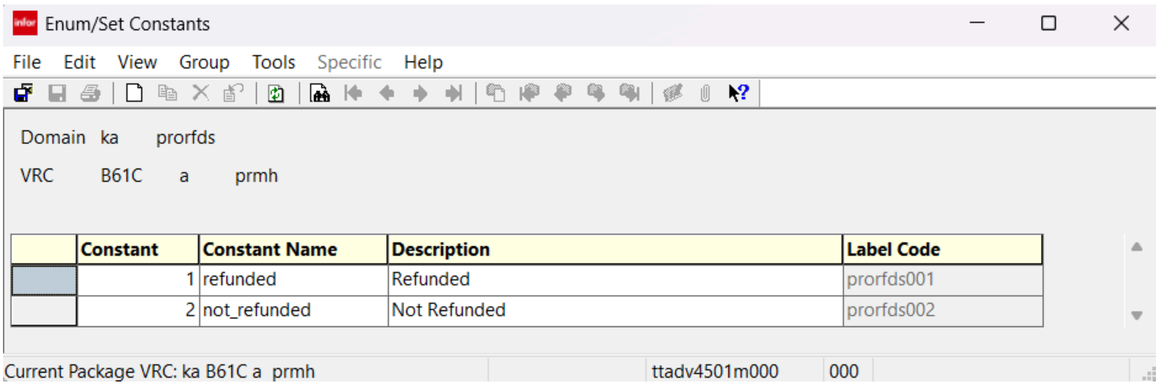
Domain ka proorig

VRC B61C a prmh

	Constant	Constant Name	Description	Label Code
	1	ord	Orders	proorig001
	2	inv	Invoice	proorig002
	3	pay	Payments	proorig003
	4	ret	Returns	proorig004
	5	buy	Buyers	proorig005
	6	sel	Sellers	proorig006
	7	prd	Products	proorig007

Current Package VRC: ka B61C a prmh ttadv4501m000 000

Refund Status:



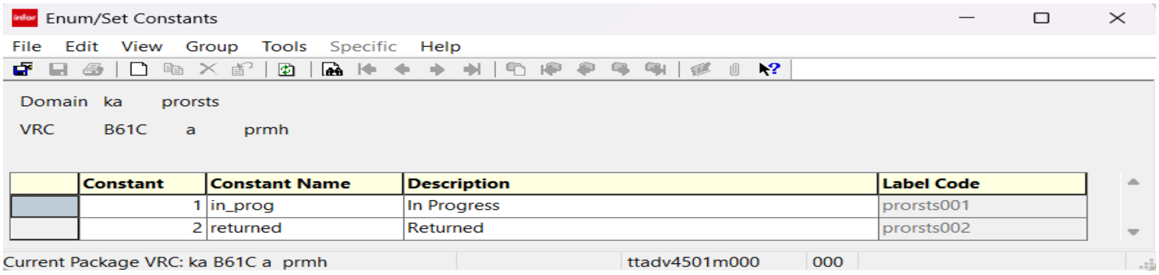
Domain ka prorfds

VRC B61C a prmh

	Constant	Constant Name	Description	Label Code
	1	refunded	Refunded	prorfds001
	2	not_refunded	Not Refunded	prorfds002

Current Package VRC: ka B61C a prmh ttadv4501m000 000

Return Status:



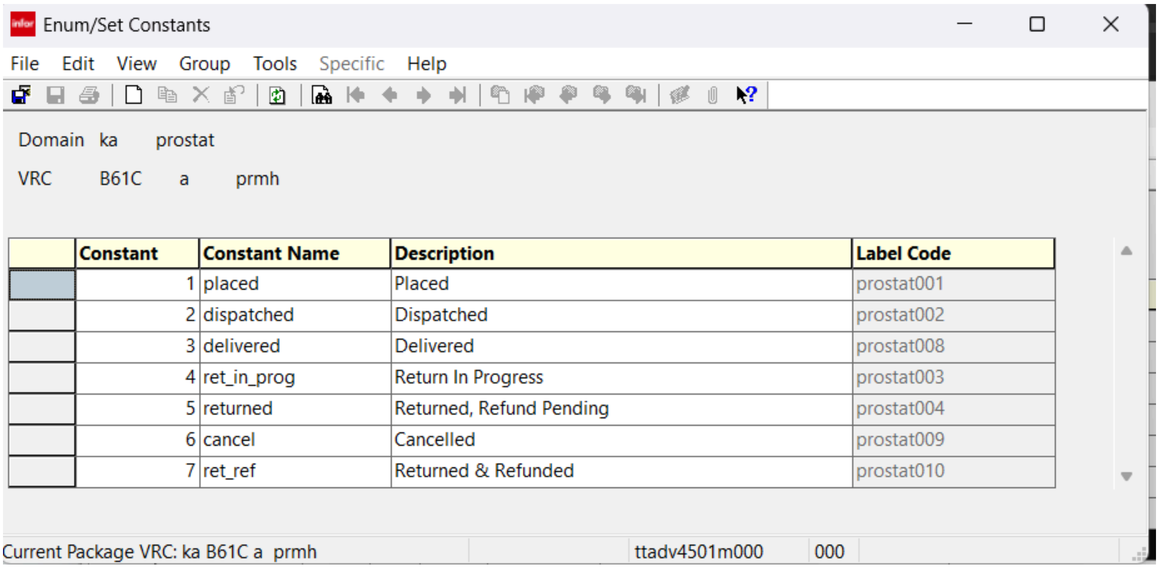
Domain ka prorsts

VRC B61C a prmh

	Constant	Constant Name	Description	Label Code
	1	in_prog	In Progress	prorsts001
	2	returned	Returned	prorsts002

Current Package VRC: ka B61C a prmh ttadv4501m000 000

Orders:



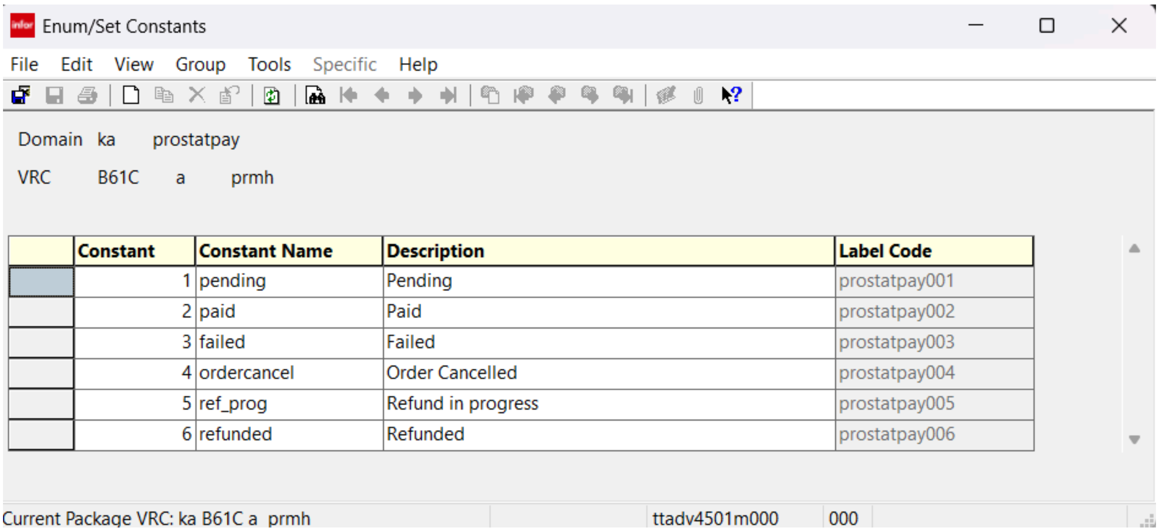
Domain ka prostat

VRC B61C a prmh

Constant	Constant Name	Description	Label Code
1	placed	Placed	prostat001
2	dispatched	Dispatched	prostat002
3	delivered	Delivered	prostat008
4	ret_in_prog	Return In Progress	prostat003
5	returned	Returned, Refund Pending	prostat004
6	cancel	Cancelled	prostat009
7	ret_ref	Returned & Refunded	prostat010

Current Package VRC: ka B61C a prmh ttadv4501m000 000

Payments:



Domain ka prostatpay

VRC B61C a prmh

Constant	Constant Name	Description	Label Code
1	pending	Pending	prostatpay001
2	paid	Paid	prostatpay002
3	failed	Failed	prostatpay003
4	ordercancel	Order Cancelled	prostatpay004
5	ref_prog	Refund in progress	prostatpay005
6	refunded	Refunded	prostatpay006

Current Package VRC: ka B61C a prmh ttadv4501m000 000

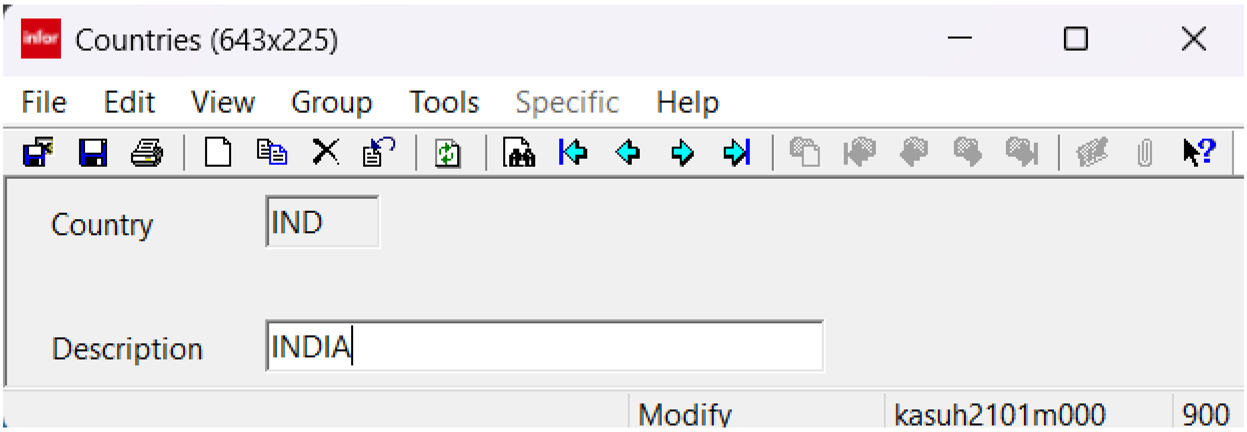
6.3 Sessions:

Sessions in Infor LN are user interfaces used to interact with tables for data entry, display, processing, or reporting.They can be of various types: **Maintain**, **Overview**, **Transaction**, and **Print** sessions.**Maintain** sessions allow CRUD operations, **Overview** sessions display records, and **Transaction** sessions handle process logic.

6.3.1 Maintain Session:

Maintain Session is used to create, update, and delete records in a specific table.It provides a form-based interface directly linked to a domain and its validation rules.This session ensures accurate data management and supports business rule enforcement.

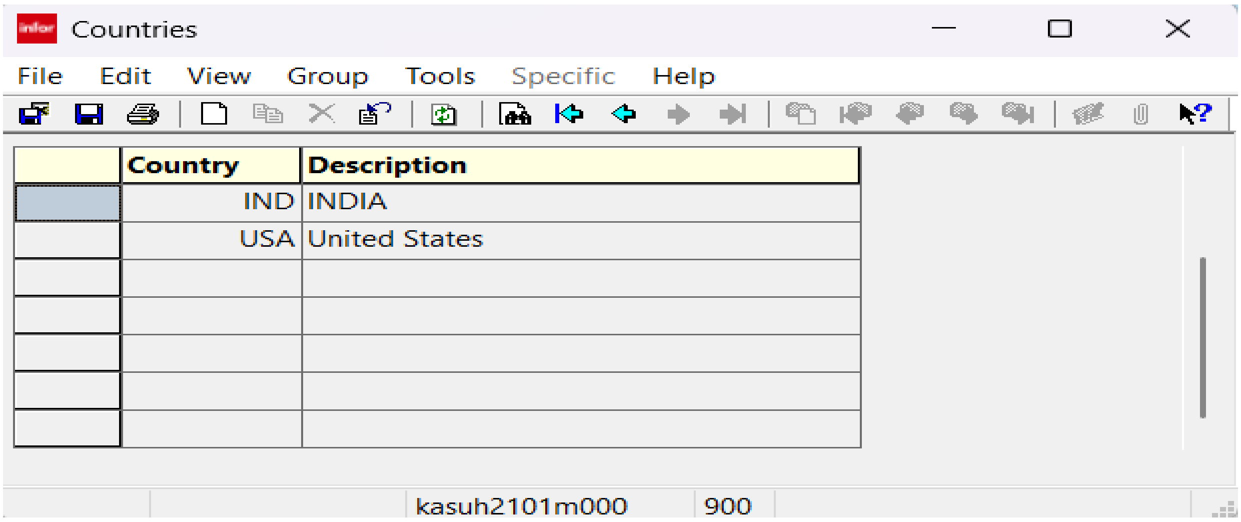
Country Detailed Session:



Country	IND
Description	INDIA

Modify kasuh2101m000 900

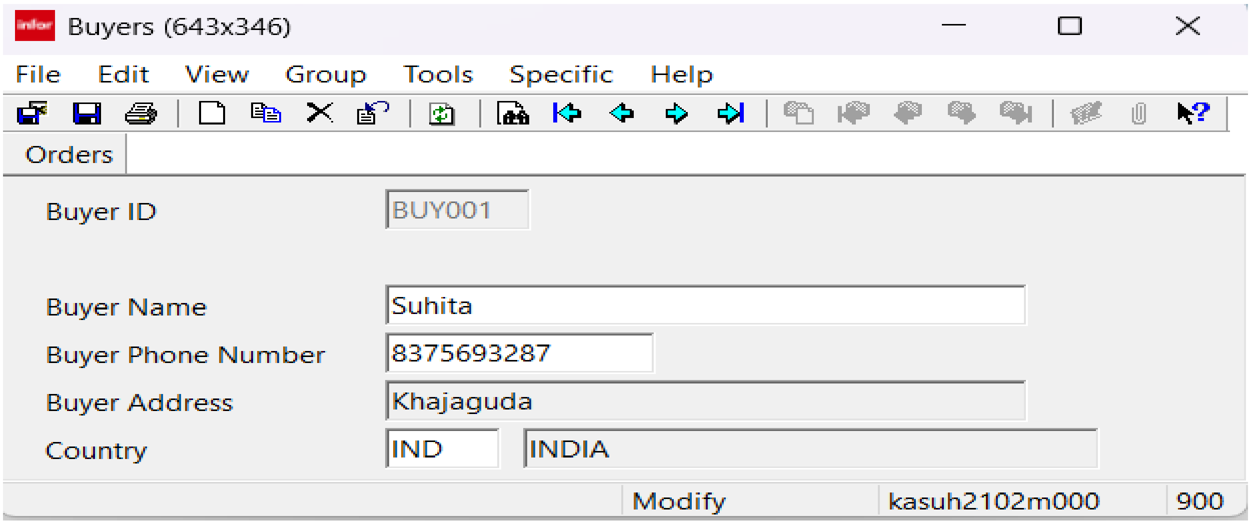
Country Overview Session:



Country	Description
IND	INDIA
USA	United States

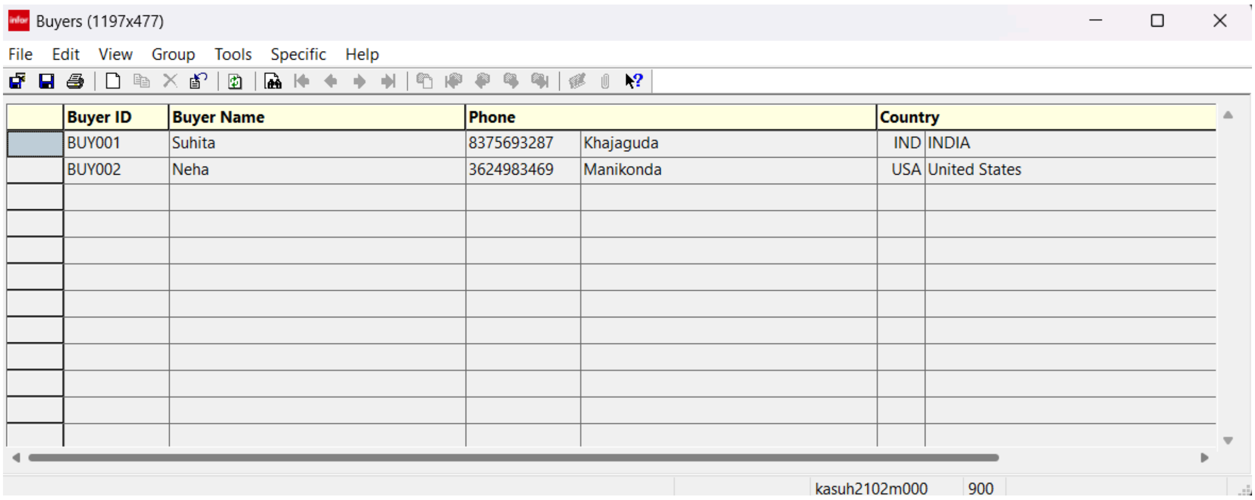
kasuh2101m000 900

Buyer Detailed Session:



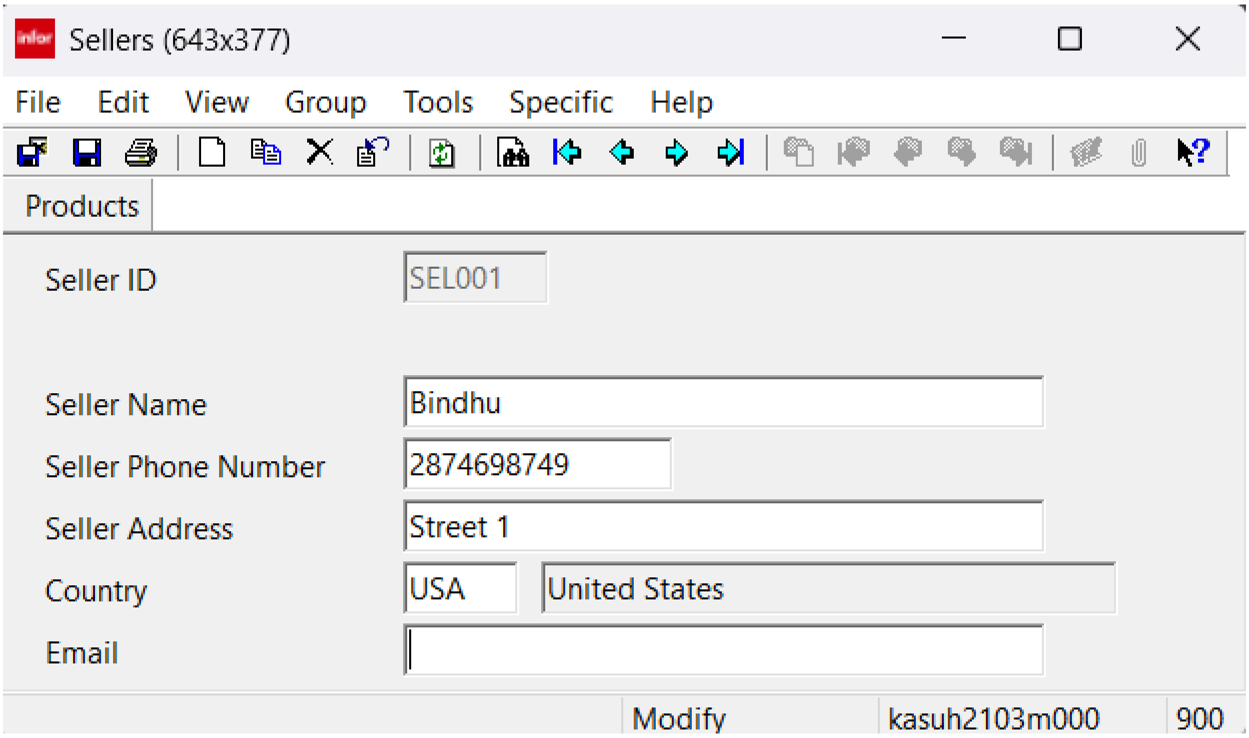
Buyer ID	BUY001	
Buyer Name	Suhita	
Buyer Phone Number	8375693287	
Buyer Address	Khajaguda	
Country	IND	INDIA

Buyer Overview Session:



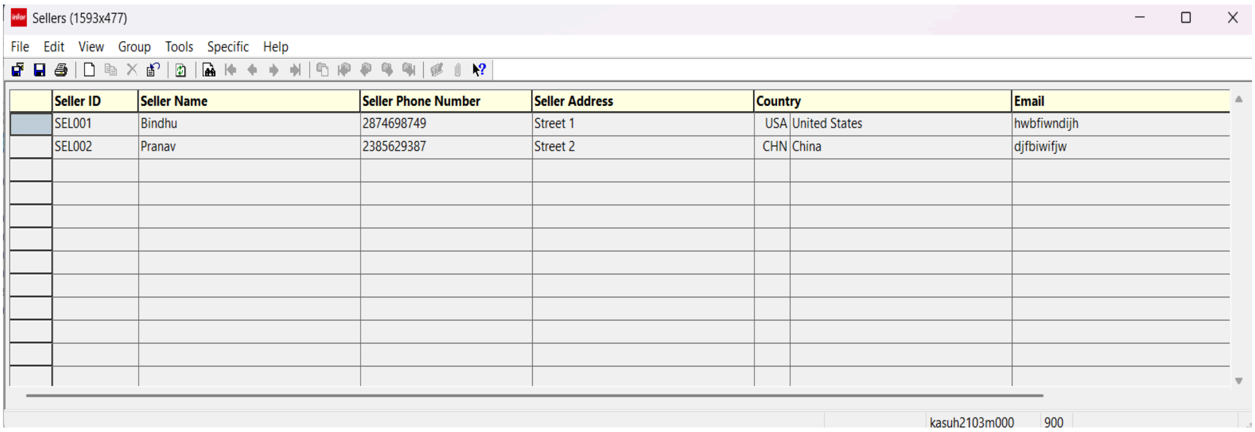
Buyer ID	Buyer Name	Phone		Country	
BUY001	Suhita	8375693287	Khajaguda	IND	INDIA
BUY002	Neha	3624983469	Manikonda	USA	United States

Seller Detailed Session:



Seller ID	SEL001	
Seller Name	Bindhu	
Seller Phone Number	2874698749	
Seller Address	Street 1	
Country	USA	United States
Email		

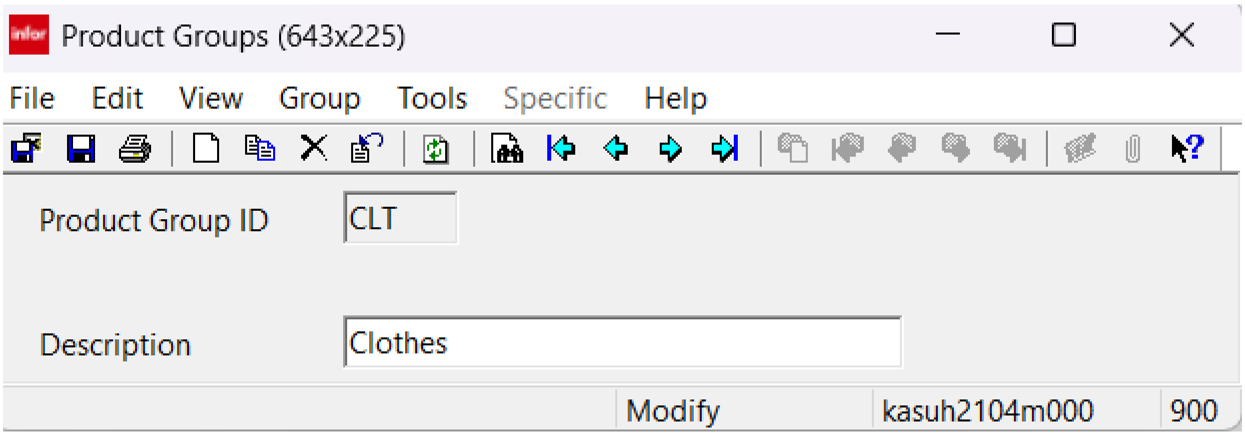
Seller Overview Session:



	Seller ID	Seller Name	Seller Phone Number	Seller Address	Country		Email
	SEL001	Bindhu	2874698749	Street 1	USA	United States	hwbfiwndijh
	SEL002	Pranav	2385629387	Street 2	CHN	China	djfbiwifjw

kasuh2103m000 900

Product Group Detailed Session:

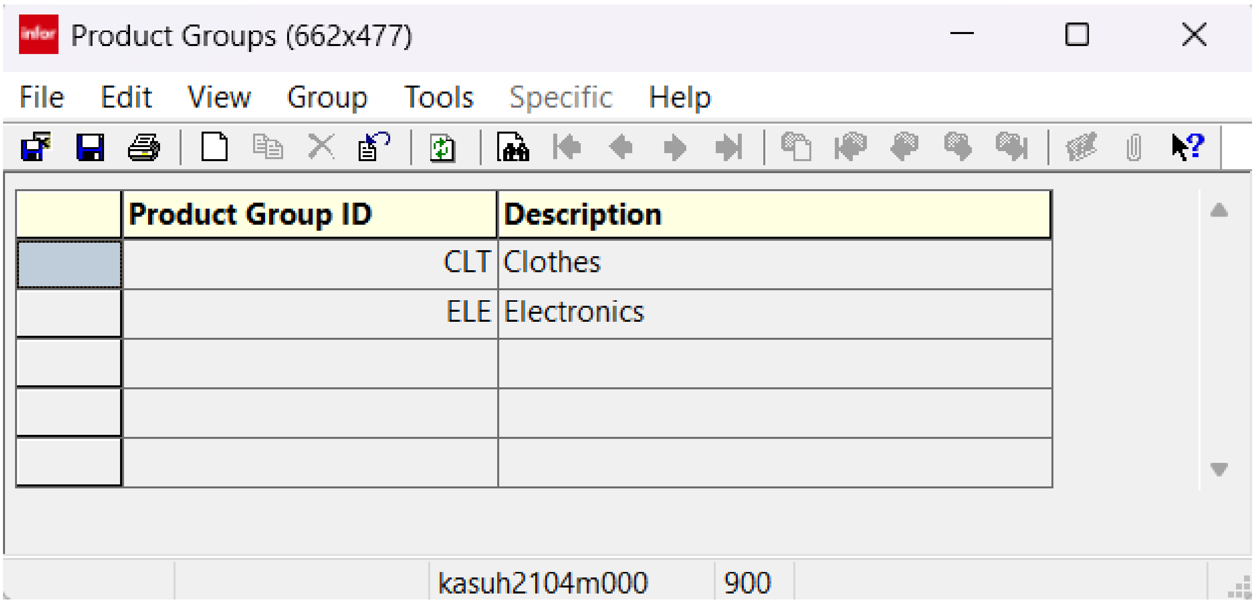


Product Group ID: CLT

Description: Clothes

Modify kasuh2104m000 900

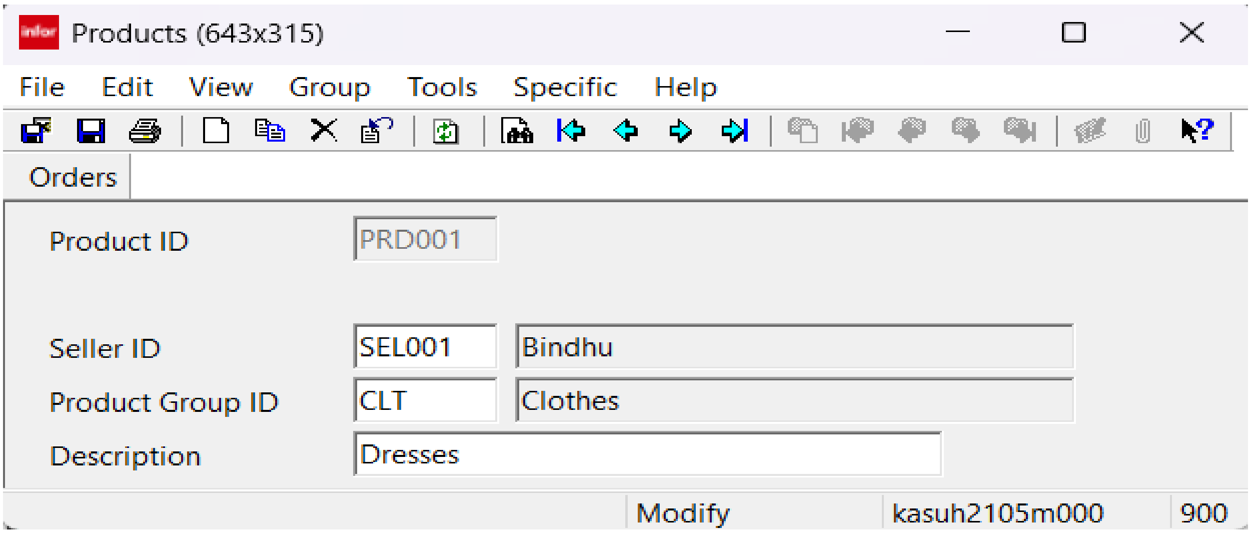
Product Group Overview Session:



	Product Group ID	Description
	CLT	Clothes
	ELE	Electronics

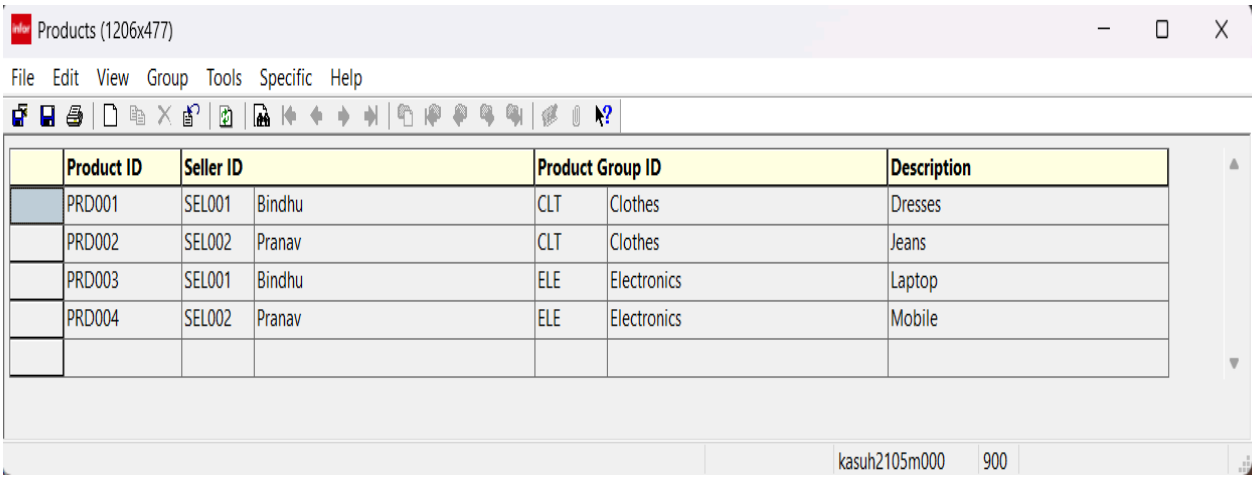
kasuh2104m000 900

Product Detailed Session:



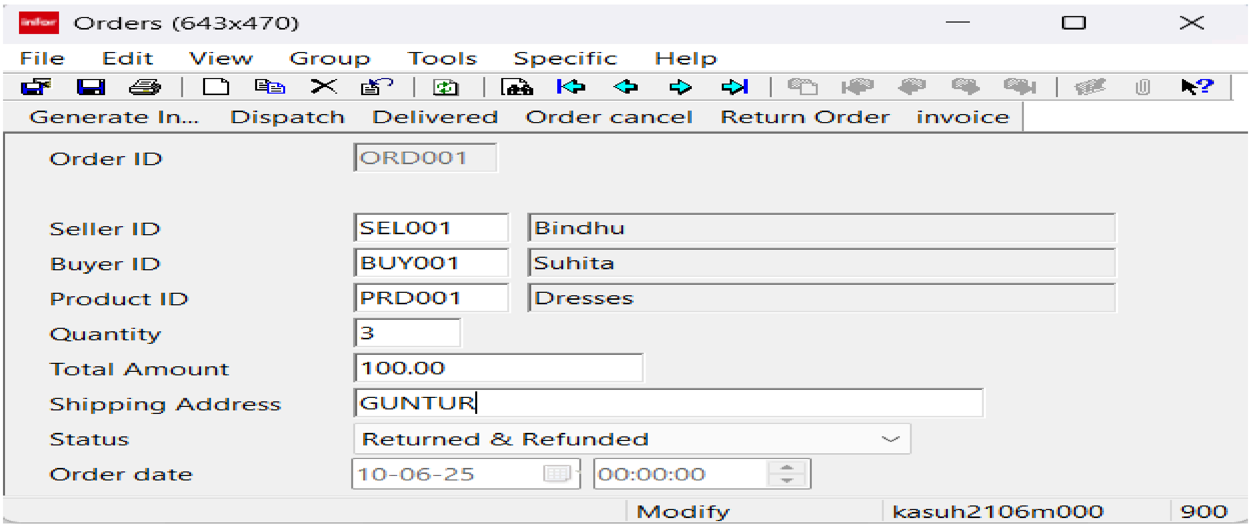
Product ID	PRD001	
Seller ID	SEL001	Bindhu
Product Group ID	CLT	Clothes
Description	Dresses	

Product Overview Session:



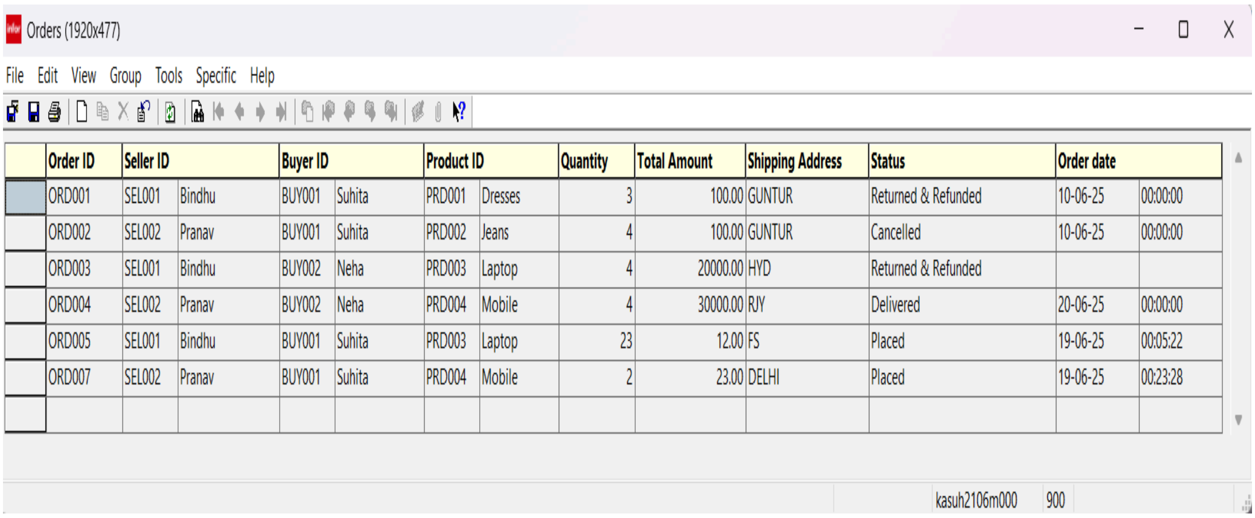
Product ID	Seller ID		Product Group ID		Description
PRD001	SEL001	Bindhu	CLT	Clothes	Dresses
PRD002	SEL002	Pranav	CLT	Clothes	Jeans
PRD003	SEL001	Bindhu	ELE	Electronics	Laptop
PRD004	SEL002	Pranav	ELE	Electronics	Mobile

Orders Detailed Session:


Orders (643x470)
File Edit View Group Tools Specific Help
Generate In... Dispatch Delivered Order cancel Return Order invoice
Modify
kasuh2106m000
900

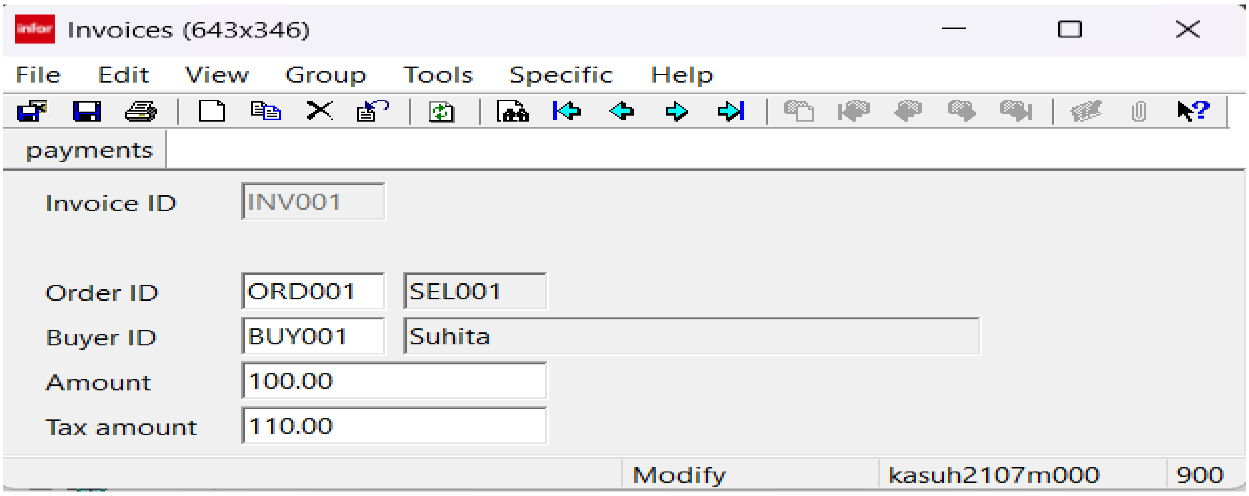
Order ID	ORD001	
Seller ID	SEL001	Bindhu
Buyer ID	BUY001	Suhita
Product ID	PRD001	Dresses
Quantity	3	
Total Amount	100.00	
Shipping Address	GUNTUR	
Status	Returned & Refunded	
Order date	10-06-25	00:00:00

Orders Overview Session:



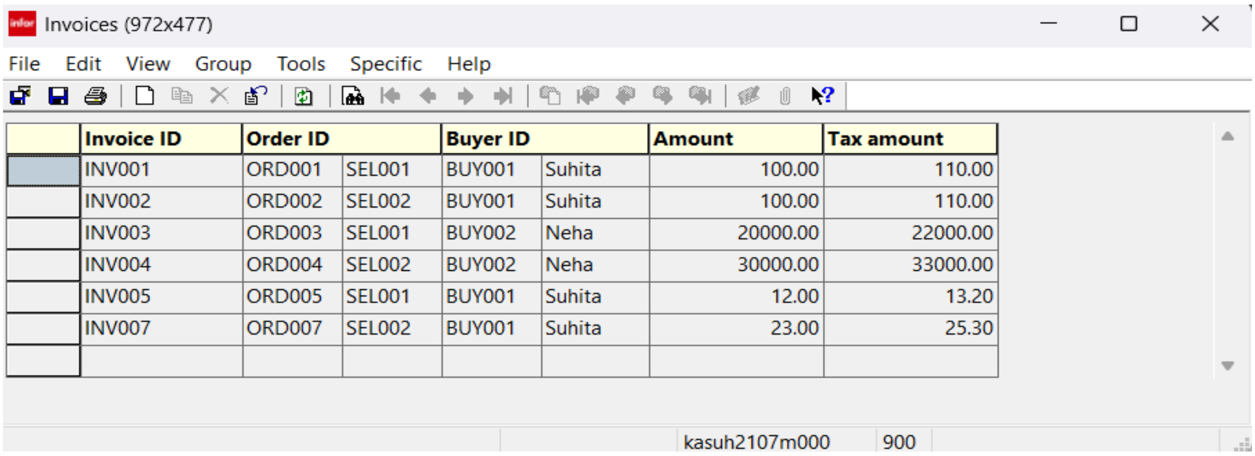
Order ID	Seller ID		Buyer ID		Product ID		Quantity	Total Amount	Shipping Address	Status	Order date	
ORD001	SEL001	Bindhu	BUY001	Suhita	PRD001	Dresses	3	100.00	GUNTUR	Returned & Refunded	10-06-25	00:00:00
ORD002	SEL002	Pranav	BUY001	Suhita	PRD002	Jeans	4	100.00	GUNTUR	Cancelled	10-06-25	00:00:00
ORD003	SEL001	Bindhu	BUY002	Neha	PRD003	Laptop	4	20000.00	HYD	Returned & Refunded		
ORD004	SEL002	Pranav	BUY002	Neha	PRD004	Mobile	4	30000.00	RJY	Delivered	20-06-25	00:00:00
ORD005	SEL001	Bindhu	BUY001	Suhita	PRD003	Laptop	23	12.00	FS	Placed	19-06-25	00:05:22
ORD007	SEL002	Pranav	BUY001	Suhita	PRD004	Mobile	2	23.00	DELHI	Placed	19-06-25	00:23:28

Invoice Detailed Session:



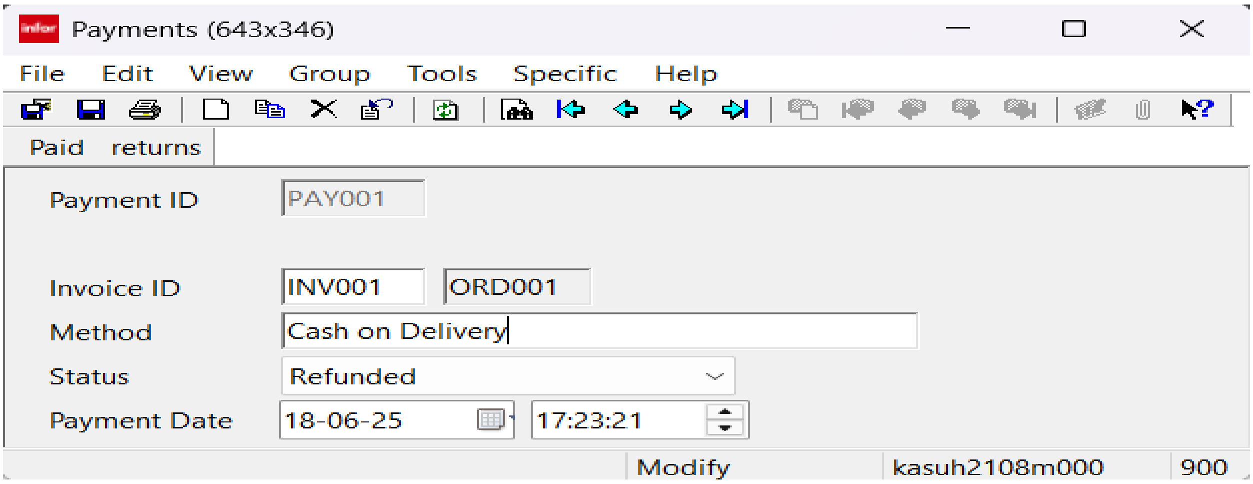
Invoice ID	INV001	
Order ID	ORD001	SEL001
Buyer ID	BUY001	Suhita
Amount	100.00	
Tax amount	110.00	

Invoice Overview Session:



Invoice ID	Order ID		Buyer ID		Amount	Tax amount
INV001	ORD001	SEL001	BUY001	Suhita	100.00	110.00
INV002	ORD002	SEL002	BUY001	Suhita	100.00	110.00
INV003	ORD003	SEL001	BUY002	Neha	20000.00	22000.00
INV004	ORD004	SEL002	BUY002	Neha	30000.00	33000.00
INV005	ORD005	SEL001	BUY001	Suhita	12.00	13.20
INV007	ORD007	SEL002	BUY001	Suhita	23.00	25.30

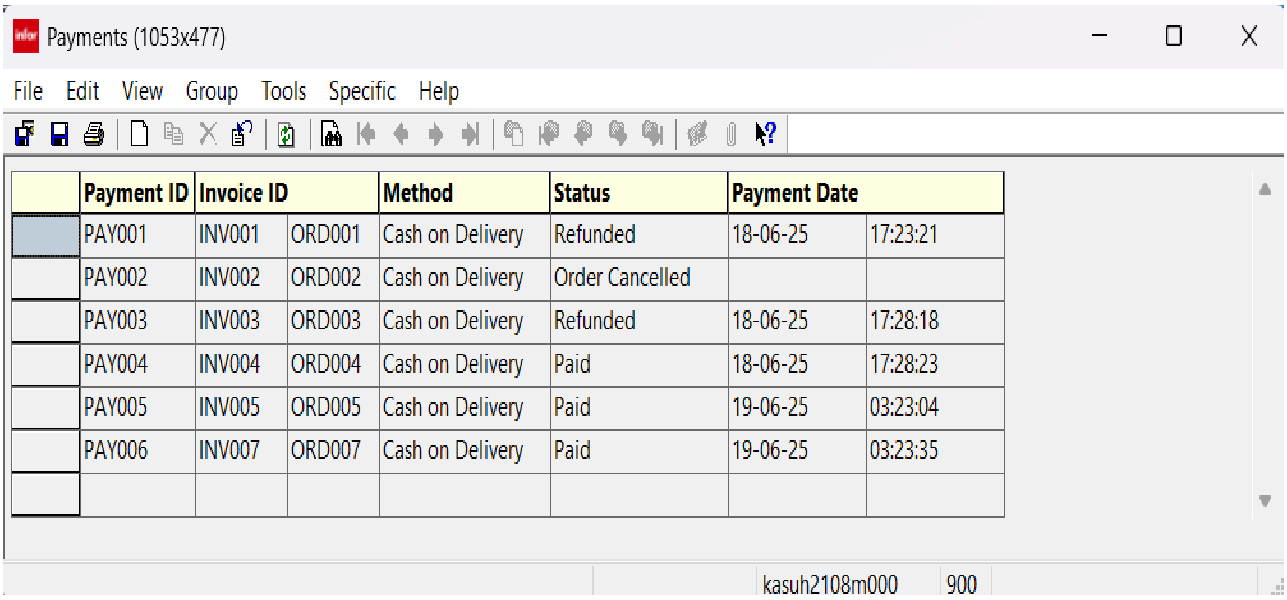
Payments Detailed Session:



Payment ID	PAY001	
Invoice ID	INV001	ORD001
Method	Cash on Delivery	
Status	Refunded	
Payment Date	18-06-25	17:23:21

kasuh2108m000 900

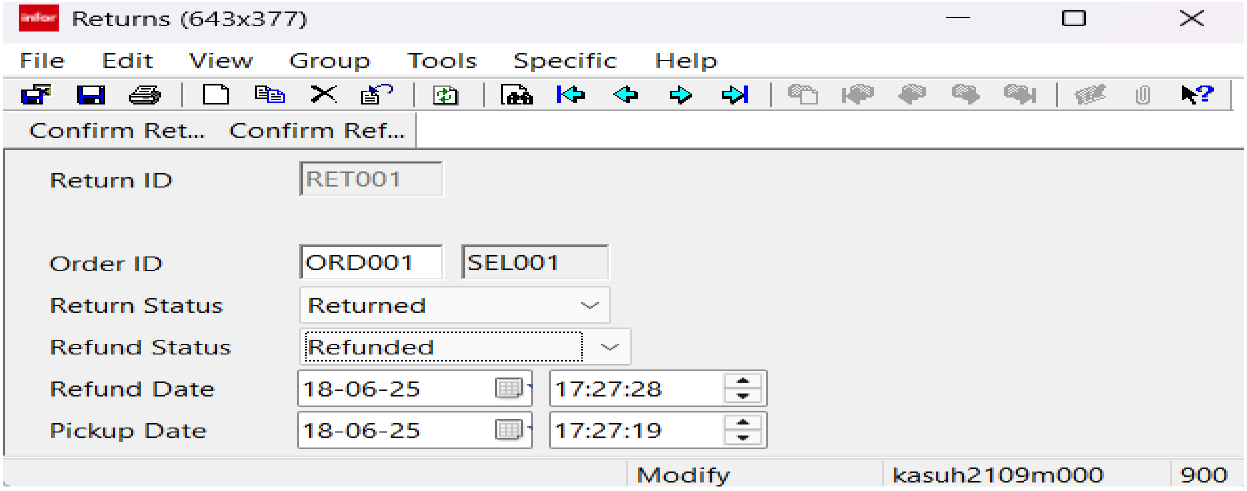
Payments Overview Session:



Payment ID	Invoice ID		Method	Status	Payment Date	
PAY001	INV001	ORD001	Cash on Delivery	Refunded	18-06-25	17:23:21
PAY002	INV002	ORD002	Cash on Delivery	Order Cancelled		
PAY003	INV003	ORD003	Cash on Delivery	Refunded	18-06-25	17:28:18
PAY004	INV004	ORD004	Cash on Delivery	Paid	18-06-25	17:28:23
PAY005	INV005	ORD005	Cash on Delivery	Paid	19-06-25	03:23:04
PAY006	INV007	ORD007	Cash on Delivery	Paid	19-06-25	03:23:35

kasuh2108m000 900

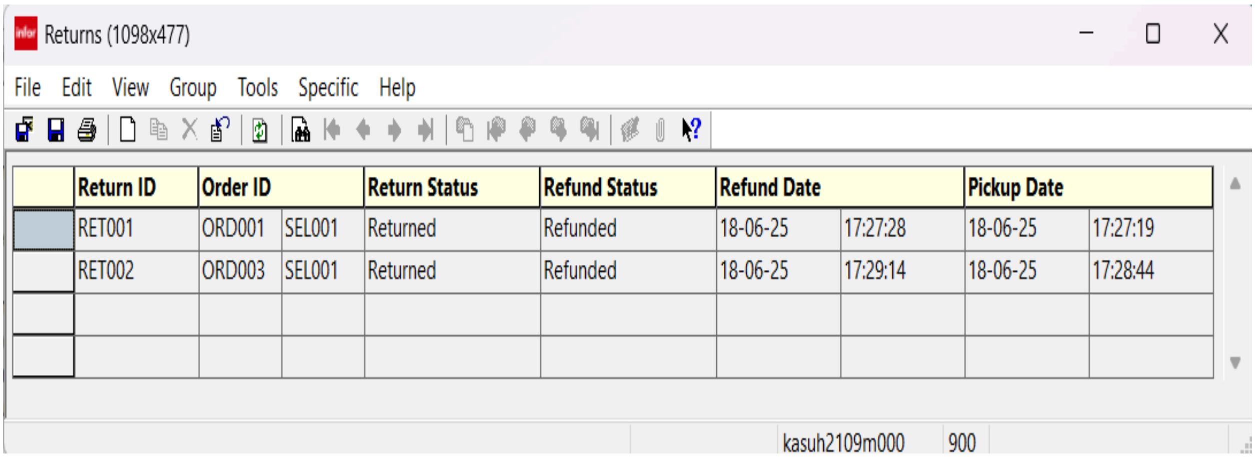
Return Detailed Session:


Returns (643x377)
File Edit View Group Tools Specific Help
Confirm Ret... Confirm Ref...
Modify

Return ID	RET001	
Order ID	ORD001	SEL001
Return Status	Returned	
Refund Status	Refunded	
Refund Date	18-06-25	17:27:28
Pickup Date	18-06-25	17:27:19

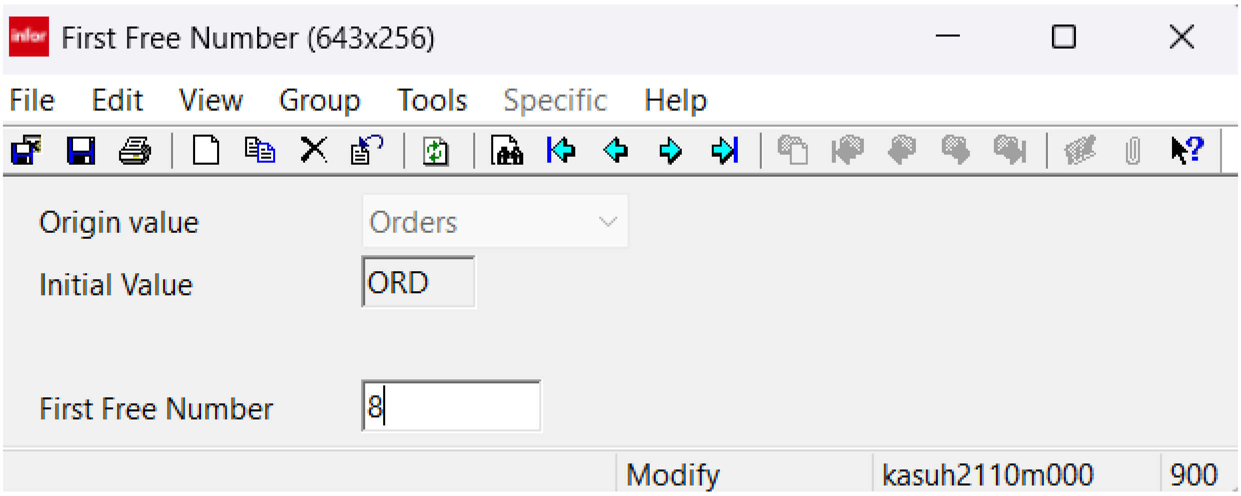
kasuh2109m000 900

Return Overview Session:



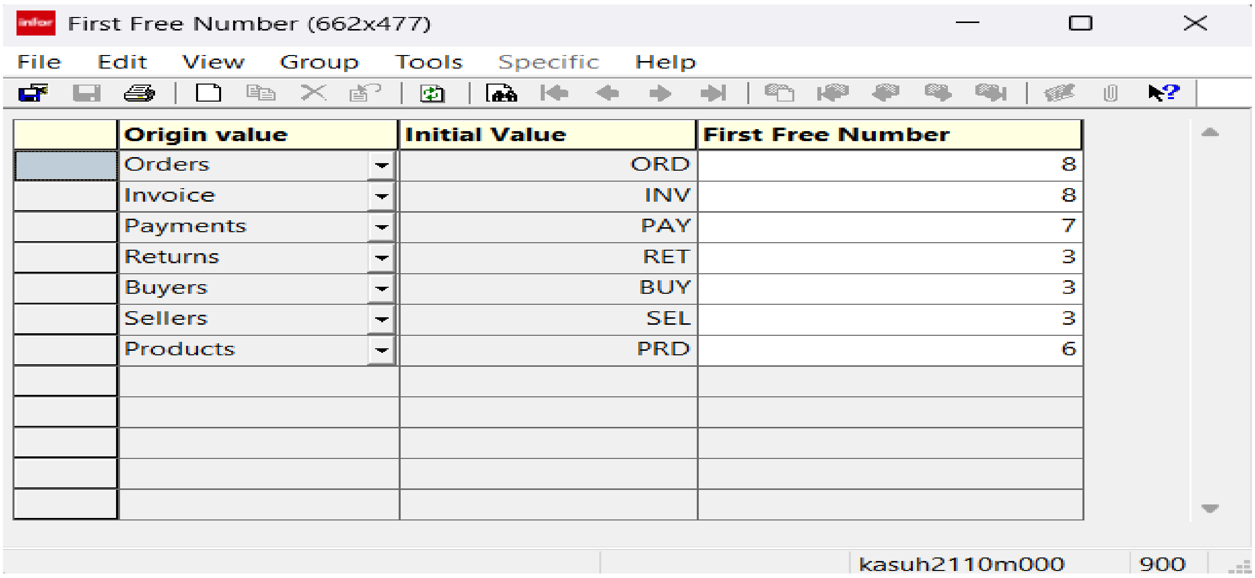
Return ID	Order ID		Return Status	Refund Status	Refund Date		Pickup Date	
RET001	ORD001	SEL001	Returned	Refunded	18-06-25	17:27:28	18-06-25	17:27:19
RET002	ORD003	SEL001	Returned	Refunded	18-06-25	17:29:14	18-06-25	17:28:44

First Free Number Detailed Session:



Origin value	Orders
Initial Value	ORD
First Free Number	8

First Free Number Overview Session:

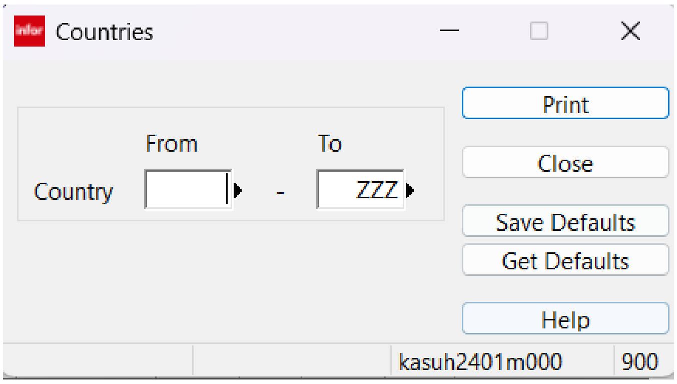


Origin value	Initial Value	First Free Number
Orders	ORD	8
Invoice	INV	8
Payments	PAY	7
Returns	RET	3
Buyers	BUY	3
Sellers	SEL	3
Products	PRD	6

6.3.2 Print Session:

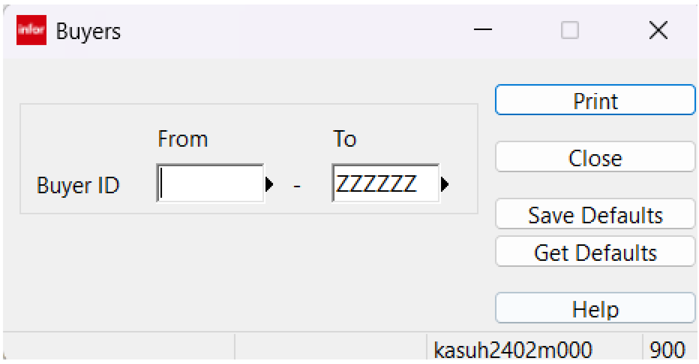
Print Session is used to generate and print reports based on table data or transaction records. It allows users to define filters, sorting options, and output formats for customized reporting. Print sessions help in reviewing, exporting, or documenting key business information efficiently.

Countries:



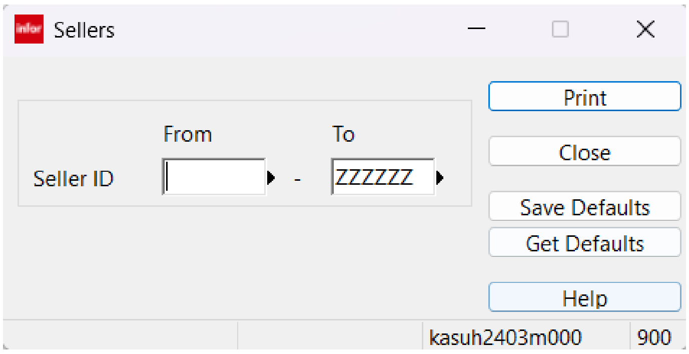
	From		To
Country		-	ZZZ

Buyers:



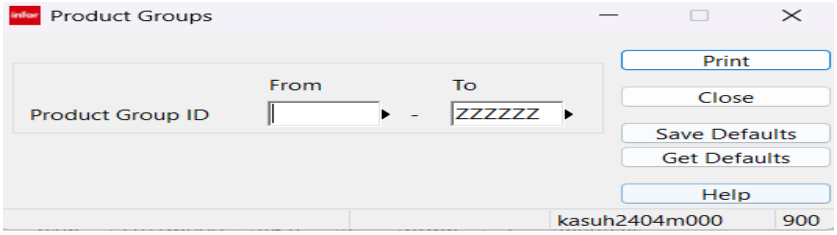
	From		To
Buyer ID		-	ZZZZZZ

Sellers:



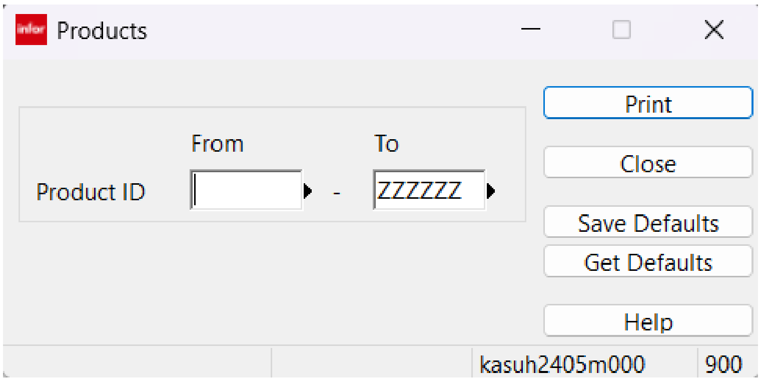
	From		To
Seller ID		-	ZZZZZZ

Product Group:



	From		To
Product Group ID		-	ZZZZZZ

Products:

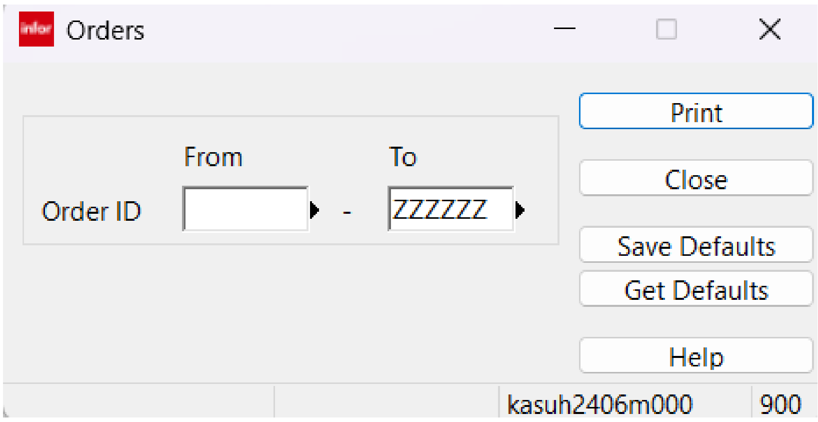


Products

	From		To
Product ID		-	ZZZZZZ

kasuh2405m000 900

Orders:

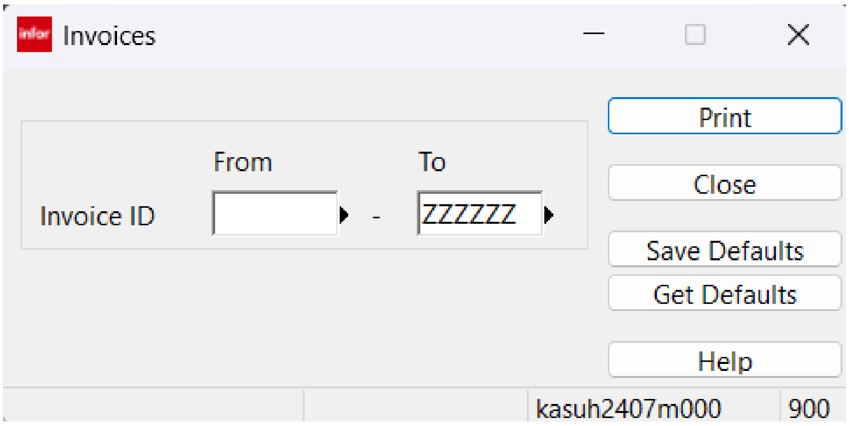


Orders

	From		To
Order ID		-	ZZZZZZ

kasuh2406m000 900

Invoice:

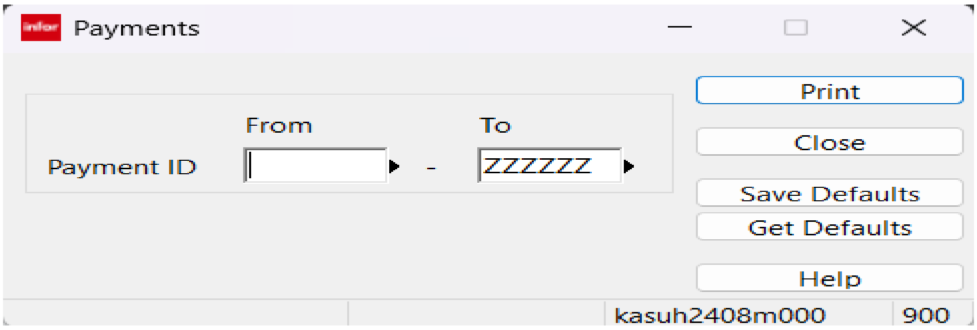


Invoices

	From		To
Invoice ID		-	ZZZZZZ

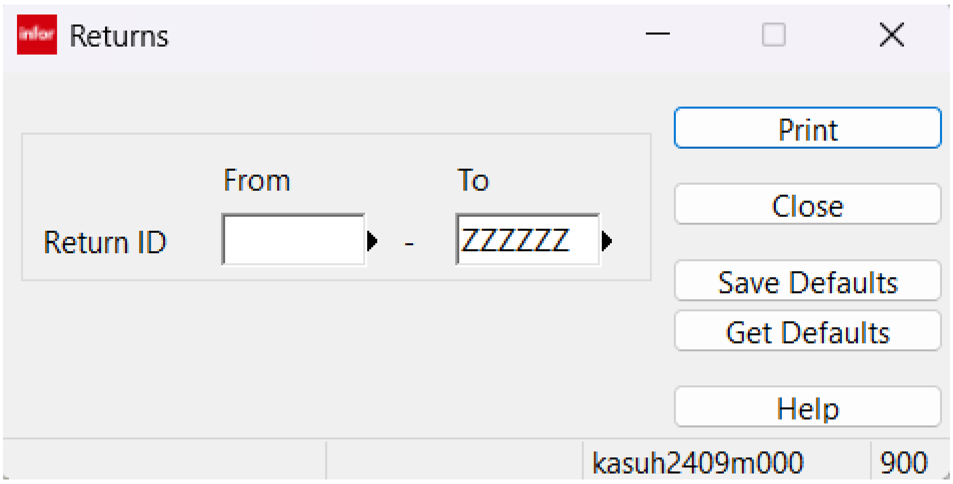
kasuh2407m000 900

Payments:


Payments

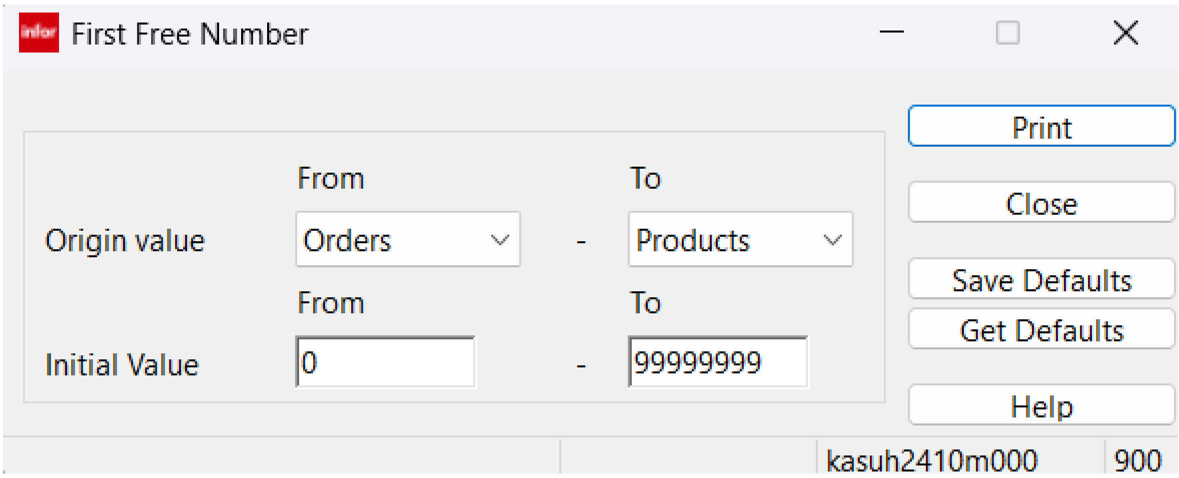
	From		To
Payment ID		-	ZZZZZZ

Returns:


Returns

	From		To
Return ID		-	ZZZZZZ

First Free Number:


First Free Number

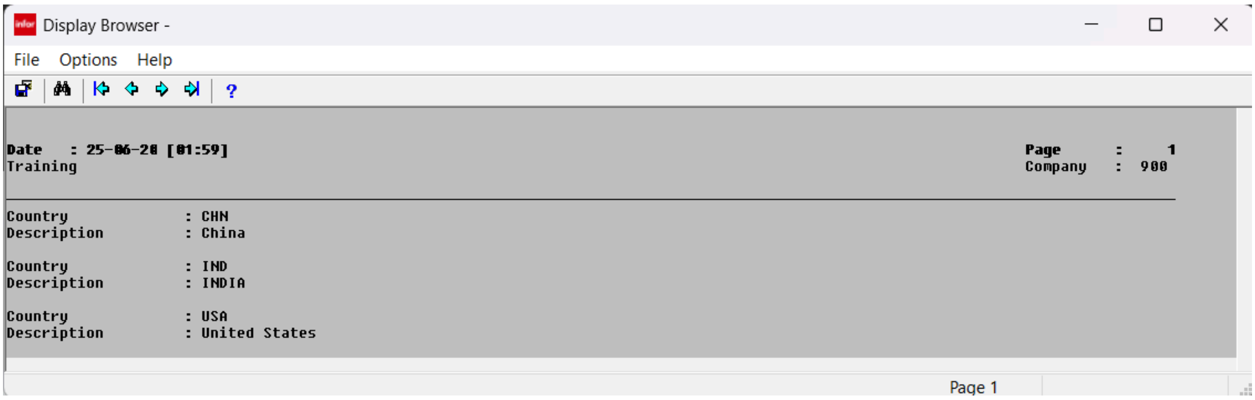
	From		To
Origin value	Orders	-	Products
Initial Value	0	-	99999999

6.4 Reports:

Reports are formatted outputs generated from application data, often used for viewing, printing, or exporting business information.A report consists of **layouts** that define the structure of the report—such as headers, footers, and body sections.

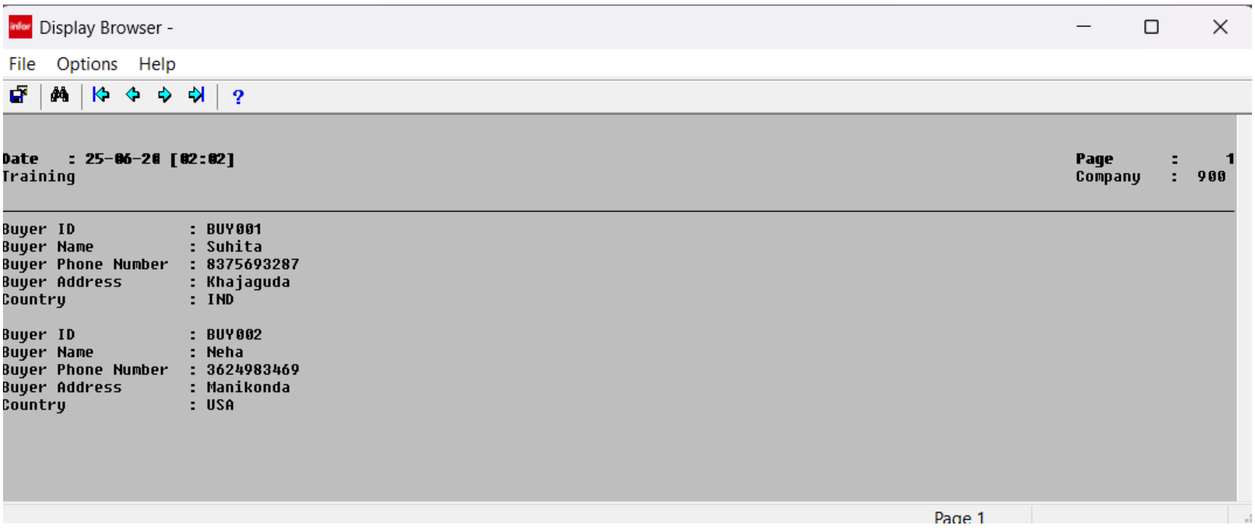
Reports can be executed via **Print Sessions**, allowing users to apply filters and view information in a structured, readable format.

Countries:



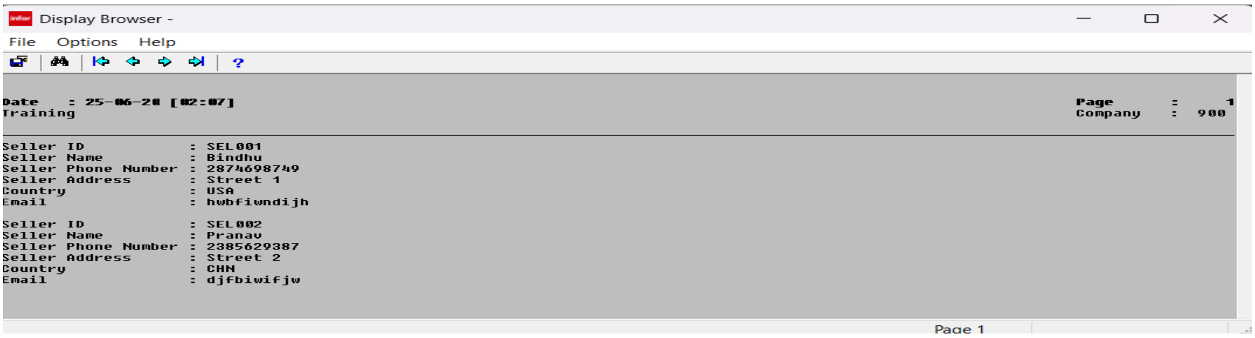
```
Date    : 25-06-28 [01:59]                                    Page     :     1
Training                                                      Company  :  900

Country              : CHN
Description          : China

Country              : IND
Description          : INDIA

Country              : USA
Description          : United States
```

Buyers:



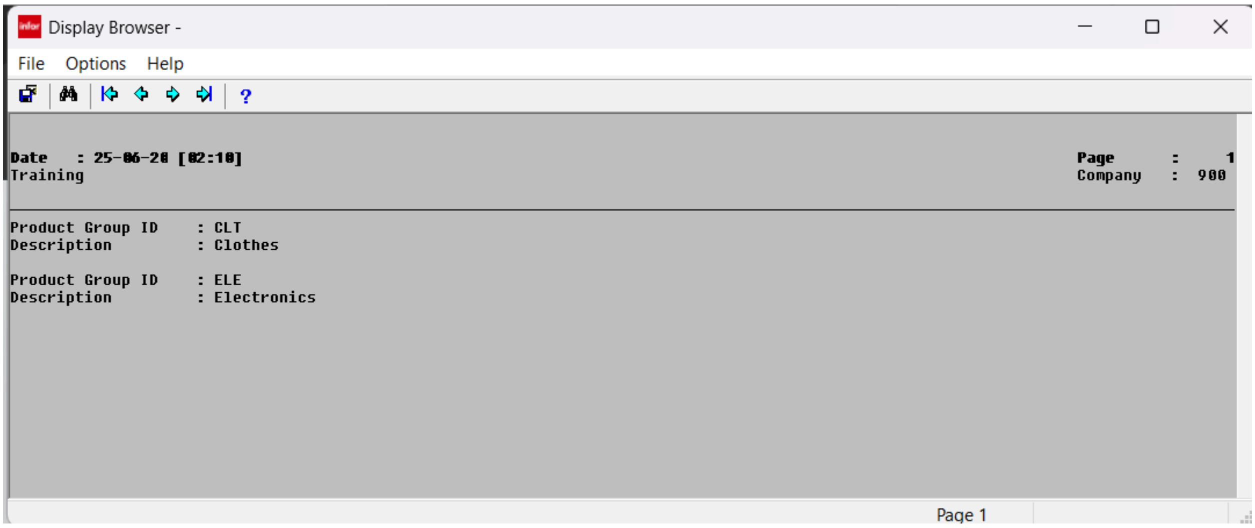
```
Date    : 25-06-28 [02:02]                                    Page     :     1
Training                                                      Company  :  900

Buyer ID             : BUY001
Buyer Name           : Suhita
Buyer Phone Number   : 8375693287
Buyer Address        : Khajaguda
Country              : IND

Buyer ID             : BUY002
Buyer Name           : Neha
Buyer Phone Number   : 3624983469
Buyer Address        : Manikonda
Country              : USA
```

Seller:



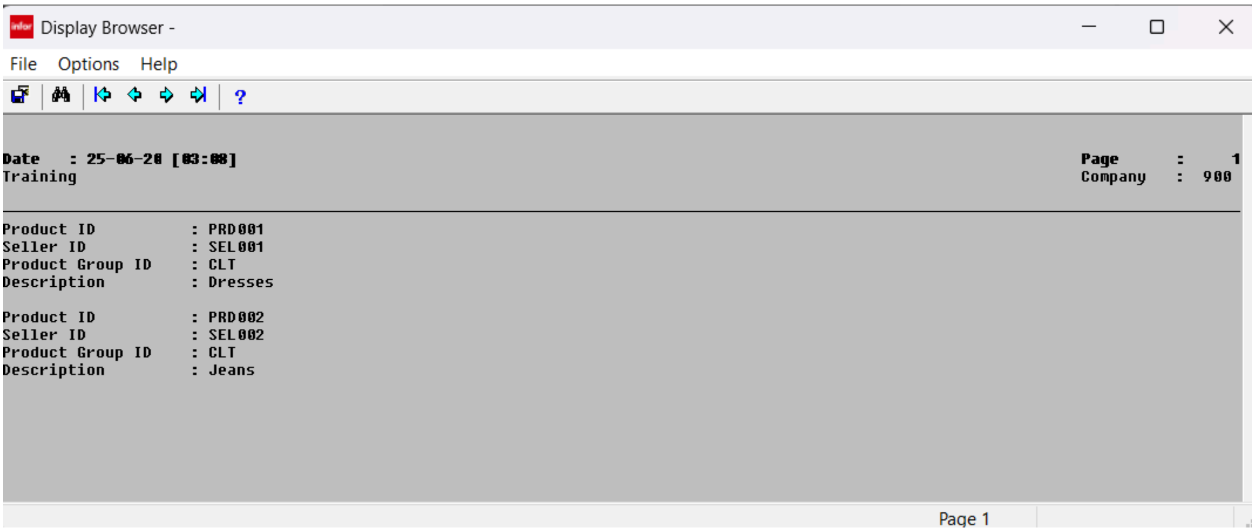
```
Date    : 25-06-28 [02:07]                                    Page     :     1
Training                                                      Company  :  900

Seller ID            : SEL001
Seller Name          : Bindhu
Seller Phone Number  : 2874698749
Seller Address       : Street 1
Country              : USA
Email                : hwbfiwndijh

Seller ID            : SEL002
Seller Name          : Pranav
Seller Phone Number  : 2385629387
Seller Address       : Street 2
Country              : CHN
Email                : djfbiwifjw
```

Product Group:



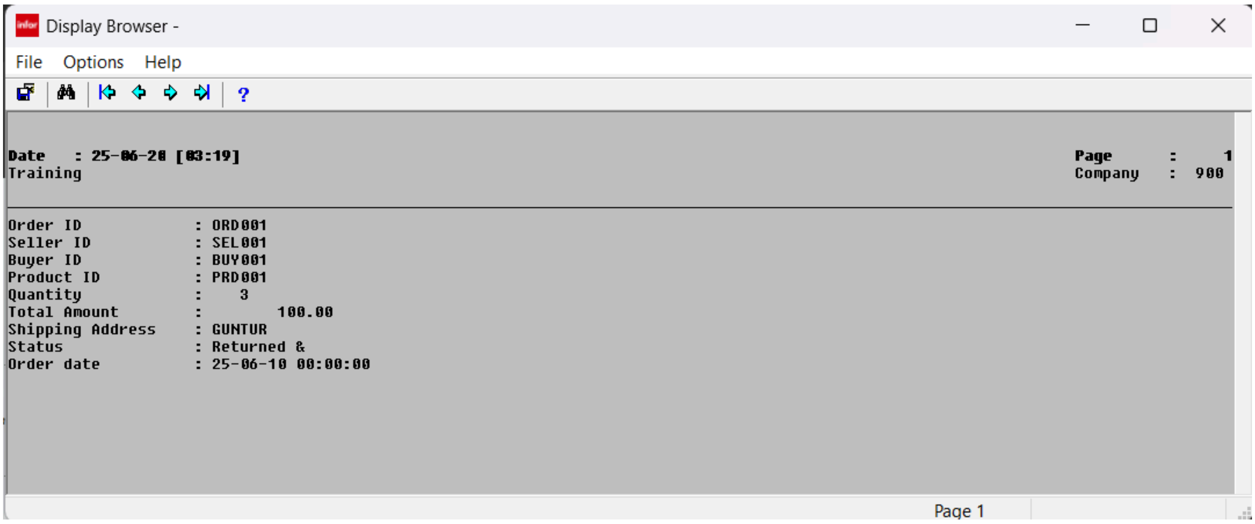
```
Date    : 25-06-28 [02:10]                                                    Page     :     1
Training                                                                      Company  :   900

Product Group ID     : CLT
Description          : Clothes

Product Group ID     : ELE
Description          : Electronics
```

Product:



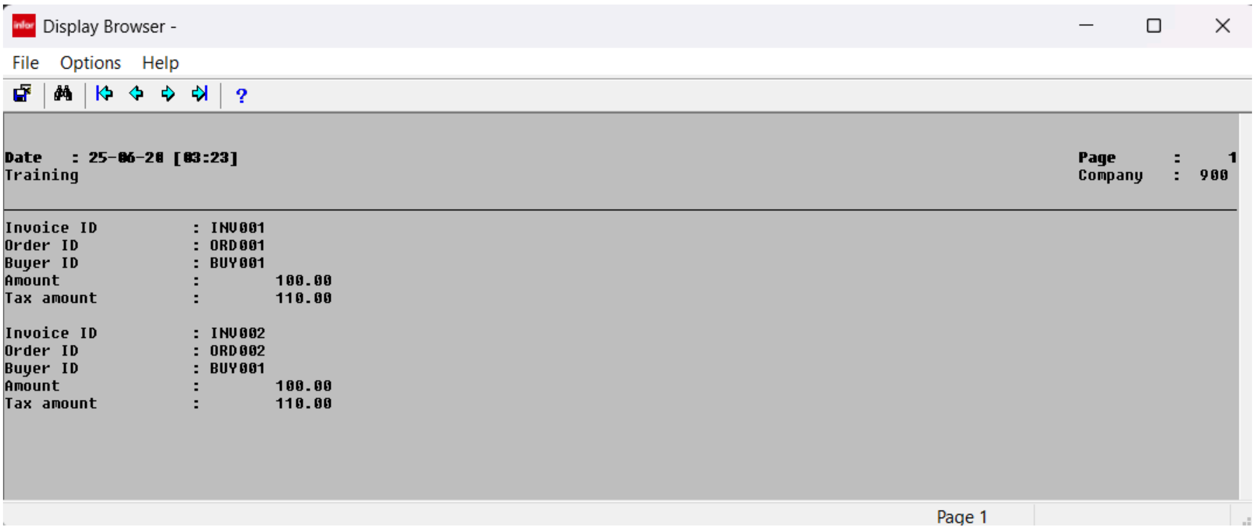
```
Date    : 25-06-28 [03:08]                                                    Page     :     1
Training                                                                      Company  :   900

Product ID            : PRD001
Seller ID             : SEL001
Product Group ID      : CLT
Description           : Dresses

Product ID            : PRD002
Seller ID             : SEL002
Product Group ID      : CLT
Description           : Jeans
```

Orders:



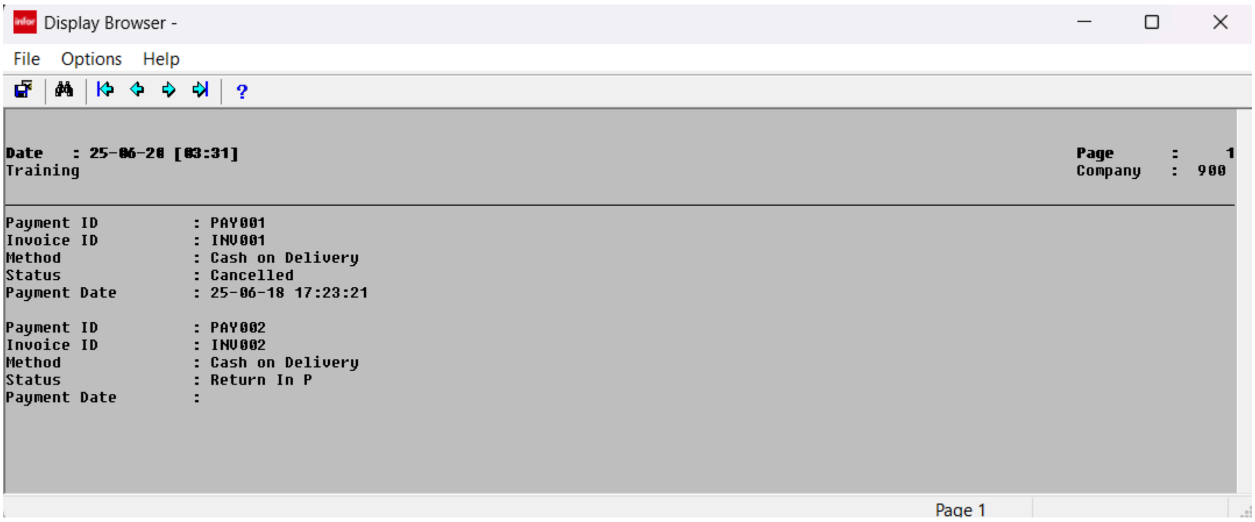
```
Date    : 25-06-28 [03:19]                                                    Page     :     1
Training                                                                      Company  :   900

Order ID              : ORD001
Seller ID             : SEL001
Buyer ID              : BUY001
Product ID            : PRD001
Quantity              :     3
Total Amount          :          100.00
Shipping Address      : GUNTUR
Status                : Returned &
Order date            : 25-06-10 00:00:00
```

Invoice:



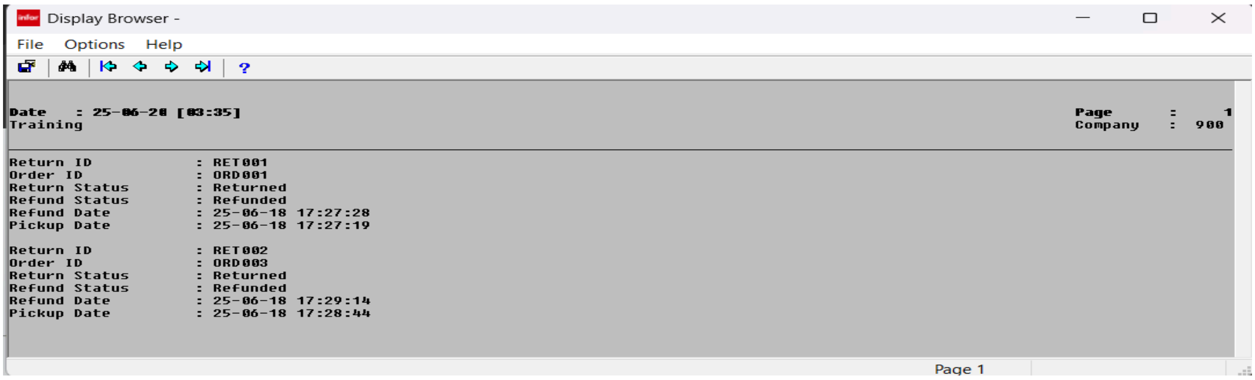
```
Date    : 25-06-20 [03:23]                                                                    Page     :     1
Training                                                                                      Company  :   900

Invoice ID          : INV001
Order ID            : ORD001
Buyer ID            : BUY001
Amount              :          100.00
Tax amount          :          110.00

Invoice ID          : INV002
Order ID            : ORD002
Buyer ID            : BUY001
Amount              :          100.00
Tax amount          :          110.00
```

Payments:



```
Date    : 25-06-20 [03:31]                                                                    Page     :     1
Training                                                                                      Company  :   900

Payment ID          : PAY001
Invoice ID          : INV001
Method              : Cash on Delivery
Status              : Cancelled
Payment Date        : 25-06-18 17:23:21

Payment ID          : PAY002
Invoice ID          : INV002
Method              : Cash on Delivery
Status              : Return In P
Payment Date        :
```

Returns:



```
Date    : 25-06-20 [03:35]                                                                    Page     :     1
Training                                                                                      Company  :   900

Return ID           : RET001
Order ID            : ORD001
Return Status       : Returned
Refund Status       : Refunded
Refund Date         : 25-06-18 17:27:28
Pickup Date         : 25-06-18 17:27:19

Return ID           : RET002
Order ID            : ORD003
Return Status       : Returned
Refund Status       : Refunded
Refund Date         : 25-06-18 17:29:14
Pickup Date         : 25-06-18 17:28:44
```

SCRIPTS

7.1 All Insertion & Deletion:

```
|****************************************************************************************
|* kasuh3glp1  0  VRC B61C a  prmh
|* All Insertion
|* tr03
|* 25-06-16 [20:47]
|****************************************************************************************
|* Script Type: 0
|****************************************************************************************


#include <bic_dialog>

extern  domain ttaad.pacc     pacc

extern  domain kapro3s                    cnty1
extern  domain kapro30s        desc1

extern  domain  kapro10s      bid1
extern  domain  kapro30s      name1
extern  domain  kapro15s      phno1
extern  domain  kapro30s      addr1
extern  domain  kapro3s       cnty2

extern  domain  kapro10s      sid1
extern  domain  kapro30s      name2
extern  domain  kapro15s      phno2
extern  domain          kapro30s        addr2
extern  domain          kapro3s                  cnty3
extern  domain          kapro30s        email

extern  domain  kapro10s      pgid1
extern  domainkapro30s        desc2

extern  domain          kapro10s        pid1
extern  domainkapro10s        sid2
extern  domainkapro10s        pgid2
extern  domainkapro30s        desc3
extern  domainkapro10s      ordid1
extern  domainkapro10s      sid3
extern  domainkapro10s      bid2
extern  domainkapro10s      pid2
extern  domainkaproint4     qty
extern  domainkaprodec      tamt1
extern  domainkapro30s      ship
extern  domainkaprostat     stat1
extern  domainkaprodate     odte

extern  domainkapro10s      invid1
extern  domainkapro10s      ordid2
extern  domainkapro10s      bid3
extern  domainkaprodec      amt
extern  domainkaprodec      tamt2

extern  domainkapro10s      payid
extern  domainkapro10s      invid2
extern  domainkapro30s      meth
extern  domainkaprostat     stat2
extern  domainkaprodate     pydt

extern  domainkapro10s      retid
extern  domainkapro10s      ordid3
extern  domainkaprorsts     rsts
extern  domainkaprorfds     rfds
extern  domainkaprodate     rdte
extern  domainkaprodate     pdte

extern  table   tkasuh201   |* Countries
extern  table   tkasuh202   |* Buyers
extern  table   tkasuh203   |* Sellers
extern  table   tkasuh204   |* Product Groups
extern  table   tkasuh205   |* Products
extern  table   tkasuh206   |* Orders
extern  table   tkasuh207   |* Invoices
extern  table   tkasuh208   |* Payments
extern  table   tkasuh209   |* Returns



extern  string  msg.text(1003)
extern  long    log.by

function main()
{
   long dlg
```

```
        msg.text = "COUNTRY."


        dlg = dialog.new("Example",
       DLG_STATUSBAR,       true,
       DLG_SAVE_GET_DFLTS,  "",
       DLG_OK_TEXT,         "All Right",
          DLG_SCROLLBARS,       true)




|*Country

dialog.add.text(dlg, msg.text)

dialog.add.field(dlg, "cnty1", "Country Code",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "desc1", "Country Description",
   DLG_MANDATORY,       true)

|*Buyers

dialog.add.text(dlg, "BUYERS." )

dialog.add.field(dlg, "bid1", "Buyer Id",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "name1", "Buyer Name",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "phno1", "Buyer Phone Number",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "addr1", "Buyer Address",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "cnty2", "Country",
   DLG_MANDATORY,       true)

|*Sellers

dialog.add.text(dlg, "SELLERS." )

dialog.add.field(dlg, "sid1", "Seller ID",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "name2", "Seller Name",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "phno2", "Seller Phone Number",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "addr2", "Seller Address",
   DLG_MANDATORY,       true)
```

```
dialog.add.field(dlg, "cnty3", "Country",
   DLG_MANDATORY,      true)
dialog.add.field(dlg, "email", "Email",
   DLG_MANDATORY,      true)


|*Product Groups

dialog.add.text(dlg, "PRODUCT GROUPS." )
dialog.add.field(dlg, "pgid1", "Product Group ID",
   DLG_MANDATORY,      true)
dialog.add.field(dlg, "desc2", "Description",
   DLG_MANDATORY,      true)


|*Products

dialog.add.text(dlg, "PRODUCTS." )

dialog.add.field(dlg, "pid1", "Product ID",
   DLG_MANDATORY,      true)
dialog.add.field(dlg, "sid2", "Seller ID",
   DLG_MANDATORY,      true)
dialog.add.field(dlg, "pgid2", "Product Group ID",
   DLG_MANDATORY,      true)
dialog.add.field(dlg, "desc3", "Description",
   DLG_MANDATORY,      true )


|*Orders

dialog.add.text(dlg, "ORDERS" )

dialog.add.field(dlg, "ordid1", "Order ID",
   DLG_MANDATORY,      true)
dialog.add.field(dlg, "sid3", "Seller ID",
   DLG_MANDATORY,      true)
dialog.add.field(dlg, "bid2", "Buyer ID",
   DLG_MANDATORY,      true)
dialog.add.field(dlg, "pid2", "Product ID",
   DLG_MANDATORY,      true)
dialog.add.field(dlg, "qty", "Quantity",
   DLG_MANDATORY,      true)
dialog.add.field(dlg, "tamt1", "Total Amount",
   DLG_MANDATORY,      true)
dialog.add.field(dlg, "ship", "Shipping Address",
   DLG_MANDATORY,      true)
dialog.add.field(dlg, "stat1", "Status",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "odte", "Order date",
   DLG_MANDATORY,       true,DLG_FIELD_TYPE,DLG_TYPE_UTC)

|*Invoices

dialog.add.text(dlg, "INVOICES" )

dialog.add.field(dlg, "invid1", "Invoice ID",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "ordid2", "Order ID",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "bid3", "Buyer ID",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "amt", "Amount",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "tamt2", "Tax amount",
   DLG_MANDATORY,       true)

|*Payments

dialog.add.text(dlg, "PAYMENTS" )

dialog.add.field(dlg, "payid", "Payment ID",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "invid2", "Invoice ID",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "meth", "Method",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "stat2", "Status",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "pydt", "Payment Date",
   DLG_MANDATORY,       true,DLG_FIELD_TYPE,DLG_TYPE_UTC)

|*RETURNS

dialog.add.text(dlg, "RETURNS" )

dialog.add.field(dlg, "retid", "Return ID",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "ordid3", "Order ID",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "rsts", "Return Status",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "rfds", "Refund Status",
   DLG_MANDATORY,       true)
dialog.add.field(dlg, "rdte", "Refund Date",
   DLG_MANDATORY,       true,DLG_FIELD_TYPE,DLG_TYPE_UTC)
```

```
    dialog.add.field(dlg, "pdte", "Pickup Date",
        DLG_MANDATORY,       true,DLG_FIELD_TYPE,DLG_TYPE_UTC)


    dialog.add.button(dlg,"insert.sth","Insert")
    dialog.add.button(dlg,"delete.sth","Delete")


    if not dialog.show(dlg) then
        message("Closing the tab.")
    endif

}

function extern insert.sth()

{

            |*Countries
            db.retry.point()
            select   kasuh201.* from         kasuh201 where          kasuh201._index1 = :cnty1
            selectdo
                        message("already present")
            selectempty
                        kasuh201.cnty = cnty1
                        kasuh201.desc = desc1
                        message("Just about to insert")
                        db.insert(tkasuh201,db.retry,e)
                        commit_or_abort()
            endselect


            |*Buyers
            db.retry.point()
            select   kasuh202.* from         kasuh202 where          kasuh202._index1 = :bid1
            selectdo
                        message("already present")
            selectempty
                        kasuh202.bid = bid1
                        kasuh202.name = name1
                        kasuh202.phno = phno1
                        kasuh202.addr = addr1
                        kasuh202.cnty = cnty2
                        message("Just about to insert")
                        db.insert(tkasuh202,db.retry,e)
                        commit_or_abort()
            endselect
```

```
|*Sellers
db.retry.point()
select  kasuh203.* from        kasuh203 where         kasuh203._index1 = :sid1
selectdo
        message("already present")
selectempty
        kasuh203.sid = sid1
        kasuh203.name = name2
        kasuh203.phno = phno2
        kasuh203.addr = addr2
        kasuh203.cnty = cnty3
        kasuh203.email = email
        message("Just about to insert")
        db.insert(tkasuh203,db.retry,e)
        commit_or_abort()
endselect

|*Product Groups
db.retry.point()
select  kasuh204.* from        kasuh204 where         kasuh204._index1 = :pgid1
selectdo
        message("already present")
selectempty
        kasuh204.pgid = pgid1
        kasuh204.desc = desc2
        message("Just about to insert")
        db.insert(tkasuh204,db.retry,e)
        commit_or_abort()
endselect

|*Products
db.retry.point()
select  kasuh205.* from        kasuh205 where         kasuh205._index1 = :pid1
selectdo
        message("already present")
selectempty
        kasuh205.pid = pid1
        kasuh205.sid = sid2
        kasuh205.pgid = pgid2
        kasuh205.desc = desc3
        message("Just about to insert")
        db.insert(tkasuh205,db.retry,e)
        commit_or_abort()
endselect

|*orders
db.retry.point()
select  kasuh206.* from        kasuh206 where         kasuh206._index1 = :ordid1
```

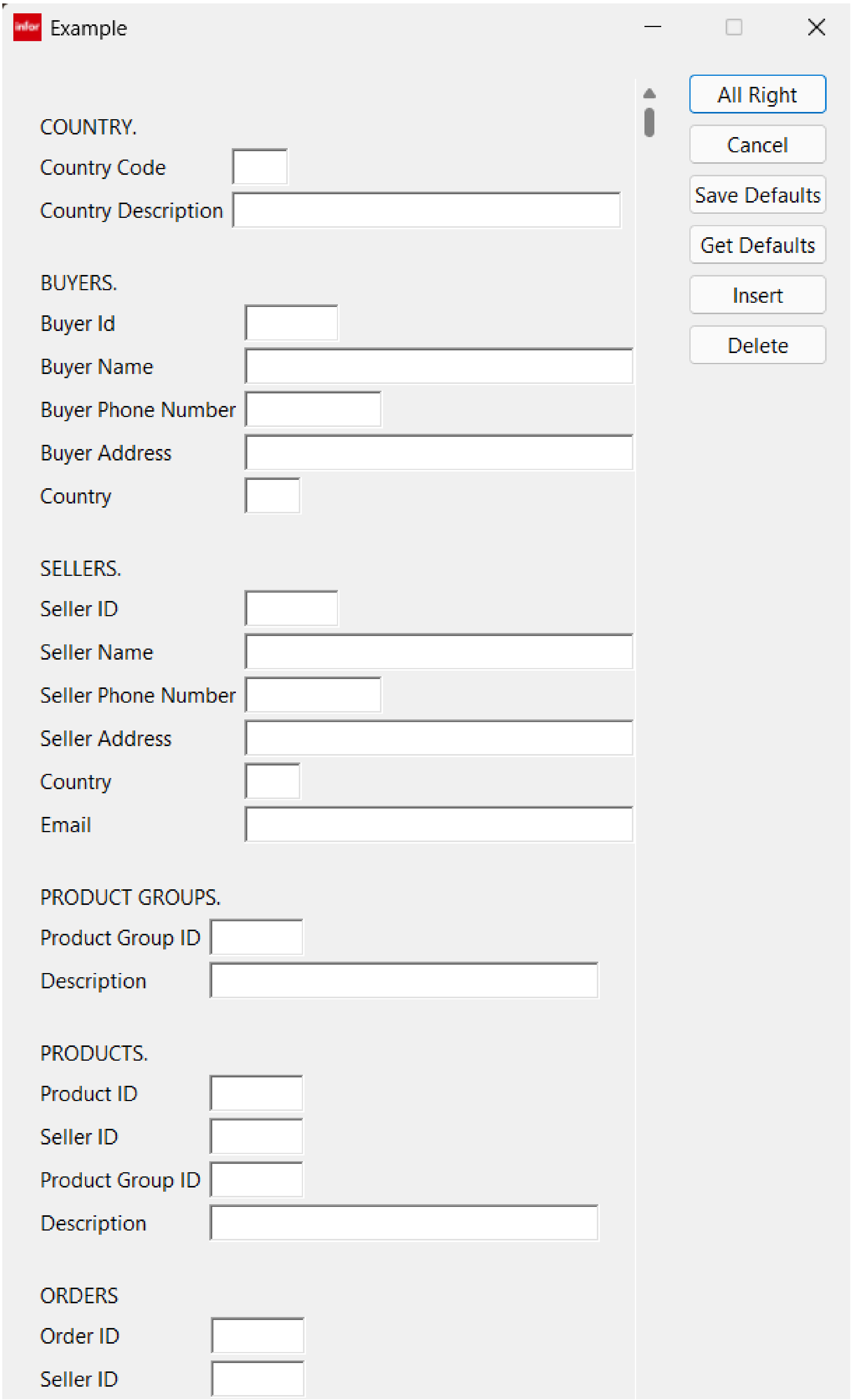
```
selectdo
        message("already present")
selectempty
        kasuh206.ordid = ordid1
        kasuh206.sid = sid3
        kasuh206.bid = bid2
        kasuh206.pid = pid2
        kasuh206.qty = qty
        kasuh206.tamt = tamt1
        kasuh206.ship = ship
        kasuh206.stat = stat1
        kasuh206.odte = odte
        message("Just about to insert")
        db.insert(tkasuh206,db.retry,e)
        commit_or_abort()
endselect

|*Invoices
db.retry.point()
select  kasuh207.* from      kasuh207 where       kasuh207._index1 = :invid1
selectdo
        message("already present")
selectempty
        kasuh207.invid = invid1
        kasuh207.ordid = ordid2
        kasuh207.bid = bid3
        kasuh207.amt = amt
        kasuh207.tamt = tamt2
        message("Just about to insert")
        db.insert(tkasuh207,db.retry,e)
        commit_or_abort()
endselect

|*Payments
db.retry.point()
select  kasuh208.* from      kasuh208 where       kasuh208._index1 = :payid
selectdo
        message("already present")
selectempty
        kasuh208.payid = payid
        kasuh208.invid = invid2
        kasuh208.meth = meth
        kasuh208.stat = stat2
        kasuh208.pydt = pydt
        message("Just about to insert")
        db.insert(tkasuh208,db.retry,e)
        commit_or_abort()
endselect
```

```
	|*Returns
	db.retry.point()
	select   kasuh209.* from        kasuh209 where          kasuh209._index1 = :retid
	as set with 1 rows

	selectdo
		message("already present")
	selectempty
		kasuh209.retid = retid
		kasuh209.ordid = ordid3
		kasuh209.rsts = rsts
		kasuh209.rfds = rfds
		kasuh209.rdte = rdte
		kasuh209.pdte = pdte
		message("Just about to insert")
		db.insert(tkasuh209,db.retry,e)
		commit_or_abort()
	endselect



}

function extern delete.sth()
{

	|*Returns
	db.retry.point()
	select *
	from kasuh209 for update
	where kasuh209._index1 = :retid
	selectdo
		db.delete(tkasuh209,db.retry,e)
		message("Deleted from table")
		commit_or_abort()
	endselect

	|*Payments
	db.retry.point()
	select *
	from kasuh208 for update
	where kasuh208._index1 = :payid
	selectdo
		db.delete(tkasuh208,db.retry,e)
		message("Deleted from table")
		commit_or_abort()
	endselect
```

```
|*Invoice
db.retry.point()
select *
from kasuh207 for update
where kasuh207._index1 = :invid1
selectdo
        db.delete(tkasuh207,db.retry,e)
        message("Deleted from table")
        commit_or_abort()
endselect

|*orders
db.retry.point()
select *
from kasuh206 for update
where kasuh206._index1 = :ordid1
selectdo
        db.delete(tkasuh206,db.retry,e)
        message("Deleted from table")
        commit_or_abort()
endselect

|*Products
db.retry.point()
select *
from kasuh205 for update
where kasuh205._index1 = :pid1
selectdo
        db.delete(tkasuh205,db.retry,e)
        message("Deleted from table")
        commit_or_abort()
endselect


|*Product Groups

db.retry.point()
select *
from kasuh204 for update
where kasuh204._index1 = :pgid1
selectdo
        db.delete(tkasuh204,db.retry,e)
        message("Deleted from table")
        commit_or_abort()
endselect
```

```
        |*Sellers
        db.retry.point()
        select  *
        from    kasuh203 for update
        where kasuh203._index1 = :sid1
        selectdo
                db.delete(tkasuh203,db.retry,e)
                commit_or_abort()

        endselect

        |*Buyers
        db.retry.point()
        select  *
        from    kasuh202 for update
        where kasuh202._index1 = :bid1
        selectdo
                db.delete(tkasuh202,db.retry,e)
                commit_or_abort()

        endselect

        |*Country
        db.retry.point()
        select  *
        from    kasuh201 for update
        where kasuh201._index1 = :cnty1
        selectdo
                db.delete(tkasuh201,db.retry,e)
                commit_or_abort()

        endselect
}

function commit_or_abort()
{
        if e = 0 then
                commit.transaction()
                mess("kabin0001",1)
        else
                abort.transaction()
                mess("kabin0002",1)
        endif
}
```

Script Output:



COUNTRY.

Country Code

Country Description

BUYERS.

Buyer Id

Buyer Name

Buyer Phone Number

Buyer Address

Country

SELLERS.

Seller ID

Seller Name

Seller Phone Number

Seller Address

Country

Email

PRODUCT GROUPS.

Product Group ID

Description

PRODUCTS.

Product ID

Seller ID

Product Group ID

Description

ORDERS

Order ID

Seller ID

7.2 All Deletion:

```
|***************************************************************************
|* kasuh3glp2  0  VRC B61C a  prmh
|* all deletion
|* tr03
|* 25-06-17 [03:45]
|***************************************************************************
|* Script Type: 0
|***************************************************************************



function main()
{
        extern  table   tkasuh201       |* Countries
        extern  table  tkasuh202        |* Buyers
        extern  table  tkasuh203        |* Sellers
        extern  table  tkasuh204        |* Product Groups
        extern  table  tkasuh205        |* Products
        extern  table  tkasuh206        |* Orders
        extern  table  tkasuh207        |* Invoices
        extern  table  tkasuh208        |* Payments
        extern  table  tkasuh209        |* Returns

        |* Step 1:
        db.retry.point()
        select * from kasuh209 for update
        selectdo
                db.delete(tkasuh209, db.retry, e)
                handle.delete.result()
        endselect

        |* Step 2:
        db.retry.point()
        select * from kasuh208 for update
        selectdo
                db.delete(tkasuh208, db.retry, e)
                handle.delete.result()
        endselect
```

```
|* Step 3:
db.retry.point()
select * from kasuh207 for update
selectdo
        db.delete(tkasuh207, db.retry, e)
        handle.delete.result()
endselect

|* Step 4:
db.retry.point()
select * from kasuh206 for update
selectdo
        db.delete(tkasuh206, db.retry, e)
        handle.delete.result()
endselect

|* Step 5:
db.retry.point()
select * from kasuh205 for update
selectdo
        db.delete(tkasuh205, db.retry, e)
        handle.delete.result()
endselect

|* Step 6:
db.retry.point()
select * from kasuh204 for update
selectdo
        db.delete(tkasuh204, db.retry, e)
        handle.delete.result()
endselect

|* Step 7:
db.retry.point()
select * from kasuh203 for update
selectdo
        db.delete(tkasuh203, db.retry, e)
        handle.delete.result()
endselect

|* Step 8:
db.retry.point()
select * from kasuh202 for update
selectdo
        db.delete(tkasuh202, db.retry, e)
        handle.delete.result()
endselect
```

```
        |* Step 9:
        db.retry.point()
        select * from kasuh201 for update
        selectdo
                db.delete(tkasuh201, db.retry, e)
                handle.delete.result()
        endselect
}

function handle.delete.result()
{
        if e = 0 then
                commit.transaction()
                message("Record deleted successfully")
        else
                abort.transaction()
                message("Record not deleted")
        endif
}
```

7.3 Countries Program Script:

```
|********************************************************************************
|* kasuh2101  0  VRC B61C a  prmh
|* Countries
|* Joshna
|* 2025-06-16
|********************************************************************************
|* Main table kasuh201 Countries, Form Type 1
|********************************************************************************

|****************************** declaration section ****************************
declaration:

  table   tkasuh201 | Countries

before.program:
        set.synchronized.dialog("kasuh2101m000")


|****************************** program section **********************************


|****************************** group section ************************************
```

7.4 Buyers Program Script:

```
|**************************************************************************
|* kasuh0202  0  VRC B61C a  prmh
|* Buyer Master - First Free Number Logic
|* Suhita Pentakota
|* 2025-06-18
|**************************************************************************

|* Main table kasuh202 Buyers, Form Type 1
|**************************************************************************

|***************************** Declaration Section **************************

declaration:

        table   tkasuh202               |* Buyer Master
        table   tkasuh210               |* First Free Number Master

        domain kapro10s                         g.bid
        long    ret

|***************************** Program Section ******************************

before.program:
        set.synchronized.dialog("kasuh2102m000")

before.display.object:
        if fattr.occurnr = 1 then
                disable.fields("kasuh202.bid")
        endif

before.new.object:
        get.first.free.number()

after.update.db.commit:
        update.first.free.number.table()

|***************************** Field Functions Section ***********************

functions:

function extern orders()
{
        start.synchronized.child("kasuh2106m000",
                                 "kasuh202.bid",
                                 "kasuh206.bid")
}

function get.first.free.number()
{
        select  kasuh210.*
        from    kasuh210
        where   kasuh210.orig = kaproorig.buy
        as set with 1 rows
        selectdo
                kasuh202.bid = kasuh210.ival & edit$(kasuh210.ffnu, "999")
        endselect
}

function update.first.free.number.table()
{
        db.retry.point()
        select  kasuh210.*
        from    kasuh210 for update
        where   kasuh210.orig = kaproorig.buy
        as set with 1 rows
        selectdo
                kasuh210.ffnu = kasuh210.ffnu + 1
                ret = db.update(tkasuh210, db.retry)
                check.and.commit.or.abort.transaction()
        endselect
}

function check.and.commit.or.abort.transaction()
{
        if ret = 0 then
                commit.transaction()
        else
                abort.transaction()
        endif
}
```

7.5 Sellers Program Script:

```
|******************************************************************************
|* kasuh0203  0  VRC B61C a  prmh
|* Seller Master - First Free Number Logic
|* Suhita Pentakota
|* 2025-06-18
|******************************************************************************

|***************************** Declaration Section **************************

declaration:

        table   tkasuh203               |* Seller Master
        table   tkasuh210               |* First Free Number Master

        domainkapro10s                  g.sid
        long    ret
```

```
|**************************** Program Section ******************************

before.program:
        set.synchronized.dialog("kasuh2103m000")

before.display.object:
        if fattr.occurnr = 1 then
                disable.fields("kasuh203.sid")
        endif

before.new.object:
        get.first.free.seller.id()

after.update.db.commit:
        update.first.free.number.for.seller()

|***************************** Field Functions Section ***********************

functions:


function extern products()
{
        start.synchronized.child("kasuh2105m000",
                                "kasuh203.sid",
                                "kasuh205.sid")
}

function get.first.free.seller.id()
{
        select  kasuh210.*
        from    kasuh210
        where   kasuh210.orig = kaproorig.sel
        as set with 1 rows
        selectdo
                kasuh203.sid = kasuh210.ival & edit$(kasuh210.ffnu, "999")
        endselect
}

function update.first.free.number.for.seller()
{
        db.retry.point()
        select  kasuh210.*
        from    kasuh210 for update
        where   kasuh210.orig = kaproorig.sel
        as set with 1 rows
        selectdo
```

```
                kasuh210.ffnu = kasuh210.ffnu + 1
                ret = db.update(tkasuh210, db.retry)
                check.and.commit.or.abort.transaction()
        endselect
}

function check.and.commit.or.abort.transaction()
{
        if ret = 0 then
                commit.transaction()
        else
                abort.transaction()
        endif
}
```

7.6 Product Groups Program Script:

```
|******************************************************************************
|* kasuh2104  0  VRC B61C a  prmh
|* Product Groups
|* Joshna
|* 2025-06-16
|******************************************************************************
|* Main table kasuh204 Product Groups, Form Type 1
|******************************************************************************

|****************************** declaration section **************************
declaration:

  table   tkasuh204 | Product Groups

before.program:
        set.synchronized.dialog("kasuh2104m000")


|****************************** program section ********************************


|****************************** group section **********************************
```

7.7 Products Program Script:

```
|************************************************************
|* kasuh0205  0  VRC B61C a  prmh
|* Product Master - First Free Number Logic
|* Suhita Pentakota
|* 2025-06-18
```

```
|*******************************************************************************

|*************************** Declaration Section **************************

declaration:

        table   tkasuh205               |* Product Master
        table   tkasuh210               |* First Free Number Master

        domainkapro10s                  g.pid
        long    ret

|**************************** Program Section ******************************

before.program:
        set.synchronized.dialog("kasuh2105m000")

before.display.object:
        if fattr.occurnr = 1 then
                disable.fields("kasuh205.pid")
        endif

before.new.object:
        get.first.free.product.id()

after.update.db.commit:
        update.first.free.number.for.product()

|*************************** Field Functions Section ***********************

functions:

function extern orders()
{
        start.synchronized.child("kasuh2106m000",
                                 "kasuh205.pid",
                                 "kasuh206.pid")
}

function get.first.free.product.id()
{
        select  kasuh210.*
        from    kasuh210
        where   kasuh210.orig = kaproorig.prd
        as set with 1 rows
        selectdo
                kasuh205.pid = kasuh210.ival & edit$(kasuh210.ffnu, "999")
        endselect
}

function update.first.free.number.for.product()
{
        db.retry.point()
        select  kasuh210.*
        from    kasuh210 for update
        where   kasuh210.orig = kaproorig.prd
        as set with 1 rows
        selectdo
                kasuh210.ffnu = kasuh210.ffnu + 1
                ret = db.update(tkasuh210, db.retry)
                check.and.commit.or.abort.transaction()
        endselect
}

function check.and.commit.or.abort.transaction()
{
        if ret = 0 then
                commit.transaction()
        else
                abort.transaction()
        endif
}
```

7.8 Orders Program Scripts:

```
|********************************************************************************
|* kasuh0206  0  VRC B61C a  prmh
|* Orders - First Free Number Logic
|* Suhita Pentakota
|* 2025-06-18
|********************************************************************************

|****************************** Declaration Section ***************************

declaration:

        table   tkasuh206               |* Orders Table
        table   tkasuh207               |* Invoices Table
        table   tkasuh208               |* Payments Table
        table   tkasuh209               |* Returns table
        table   tkasuh210               |* First Free Number Master

        boolean         exists
        long            ret
extern domain kapro10s          invid
```

```
|***************************** Program Section ******************************

before.program:
        set.synchronized.dialog("kasuh2106m000")

before.display.object:
        if fattr.occurnr = 1 then
                disable.fields("kasuh206.ordid")
                disable.fields("kasuh206.odte")
        endif

before.new.object:
        get.first.free.order.id()

after.update.db.commit:
        update.first.free.number.for.order()

|***************************** Field Functions Section ***********************

functions:

function extern invoice()
{
        start.synchronized.child("kasuh2107m000",
                                 "kasuh206.ordid",
                                 "kasuh207.ordid")
}


function get.first.free.order.id()
{
        select  kasuh210.*
        from    kasuh210
        where   kasuh210.orig = kaproorig.ord
        as set with 1 rows
        selectdo
                kasuh206.ordid = kasuh210.ival & edit$(kasuh210.ffnu, "999")
        endselect
}

function update.first.free.number.for.order()
{
        db.retry.point()
        select  kasuh210.*
        from    kasuh210 for update
        where   kasuh210.orig = kaproorig.ord
        as set with 1 rows
```

```
          selectdo
                    kasuh210.ffnu = kasuh210.ffnu + 1
                    ret = db.update(tkasuh210, db.retry)
                    check.and.commit.or.abort.transaction()
          endselect
}

function check.and.commit.or.abort.transaction()
{
          if ret = 0 then
                    commit.transaction()
          else
                    abort.transaction()
          endif
}

|*------------------------------------------------------------------------------------------------
function extern approve()
{
          select kasuh207.*
          from kasuh207
          where kasuh207.ordid = :kasuh206.ordid
          as set with 1 rows
          selectempty
                    update.order.confirmation.date() |*added1
                    insert.invoice.from.order()
                    update.first.free.number.for.invoice()
                    insert.default.payment()
                    update.first.free.number.for.payment()

          selectdo
                    message("Invoice already generated for selected order")
          endselect

          refresh.curr.occ()
}

|*added 2

function update.order.confirmation.date()
{
          db.retry.point()
          select kasuh206.*
          from kasuh206 for update
          where kasuh206._index1 = {:kasuh206.ordid}
          as set with 1 rows
          selectdo
                    kasuh206.odte = utc.num()
```

```
                ret = db.update(tkasuh206, db.retry)
                check.and.commit.or.abort.transaction()
        endselect
}

|*end

function insert.invoice.from.order()
{
        select kasuh210.*  |* Get Invoice first free number
        from kasuh210
        where kasuh210.orig = kaproorig.inv
        as set with 1 rows
        selectdo
                kasuh207.invid = kasuh210.ival & edit$(kasuh210.ffnu, "999")
        endselect

        kasuh207.ordid = kasuh206.ordid
        kasuh207.bid   = kasuh206.bid
        kasuh207.amt   = kasuh206.tamt
        kasuh207.tamt  = kasuh206.tamt * 1.1  |* Assume 10% tax for example

        ret = db.insert(tkasuh207, db.retry)
        check.and.commit.or.abort.transaction()
}

function update.first.free.number.for.invoice()
{
        db.retry.point()
        select kasuh210.*
        from kasuh210 for update
        where kasuh210.orig = kaproorig.inv
        as set with 1 rows
        selectdo
                kasuh210.ffnu = kasuh210.ffnu + 1
                ret = db.update(tkasuh210, db.retry)
                check.and.commit.or.abort.transaction()
        endselect
}

function insert.default.payment()
{
        select kasuh210.*  |* Get Payment first free number
        from kasuh210
        where kasuh210.orig = kaproorig.pay
        as set with 1 rows
        selectdo
                kasuh208.payid = kasuh210.ival & edit$(kasuh210.ffnu, "999")
        endselect

        kasuh208.invid = kasuh207.invid
        kasuh208.meth  = "Cash on Delivery"
        kasuh208.stat  = kaprostatpay.pending   |* Assuming default status is Pending
        kasuh208.pydt  = 0                |* Default to 0 (unset)

        ret = db.insert(tkasuh208, db.retry)
        check.and.commit.or.abort.transaction()
}

function update.first.free.number.for.payment()
{
        db.retry.point()
        select kasuh210.*
        from kasuh210 for update
        where kasuh210.orig = kaproorig.pay
        as set with 1 rows
        selectdo
                kasuh210.ffnu = kasuh210.ffnu + 1
                ret = db.update(tkasuh210, db.retry)
                check.and.commit.or.abort.transaction()
        endselect
}

|*--------------------------------------------------------------------------------------
function extern dispatch()
{
        change.status.of.order.to.dispatched()
        refresh.curr.occ()
}

function change.status.of.order.to.dispatched()
{
        db.retry.point()
        select   kasuh206.*
        from     kasuh206 for update
        where   kasuh206._index1 = {:kasuh206.ordid}
        as set with 1 rows
        selectdo
                if kasuh206.stat = kaprostat.placed then
                        kasuh206.stat = kaprostat.dispatched
                        ret = db.update(tkasuh206, db.retry)
                        check.and.commit.or.abort.transaction()
                else
                        message("Order can only be dispatched if it is currently in 'placed' status.")
                endif
        endselect
```

```
}

|*-------------------------------------------------------------------------------------------
function extern delivered()
{
    select kasuh206.stat
    from kasuh206
    where kasuh206._index1 = {:kasuh206.ordid}
    as set with 1 rows
    selectdo
        if kasuh206.stat = kaprostat.dispatched then
            mark.order.as.delivered()
            update.payment.status.for.order()
            refresh.curr.occ()
        else
            message("Order must be in 'dispatched' status to be marked as delivered.")
        endif
    endselect

}

function mark.order.as.delivered()
{
    db.retry.point()
    select kasuh206.*
    from kasuh206 for update
    where kasuh206._index1 = {:kasuh206.ordid}
    as set with 1 rows
    selectdo
        kasuh206.stat = kaprostat.delivered
        ret = db.update(tkasuh206, db.retry)
        check.and.commit.or.abort.transaction()
    endselect
}

function update.payment.status.for.order()
{
    db.retry.point()
    select kasuh207.invid
    from kasuh207
    where kasuh207.ordid = :kasuh206.ordid
    as set with 1 rows
    selectdo
        select kasuh208.*
        from kasuh208 for update
        where kasuh208.invid = {:kasuh207.invid}
        as set with 1 rows
        selectdo
```

```
            kasuh208.stat = kaprostatpay.paid
            kasuh208.pydt = utc.num()
            ret = db.update(tkasuh208, db.retry)
{           check.and.commit.or.abort.transaction()
        endselect
    endselect
}
|*---------------------------------------------------------------------------
function extern ordercancel()
{
    db.retry.point()
    select  kasuh206.*
    from    kasuh206 for update
    where   kasuh206._index1 = {:kasuh206.ordid}
    as set with 1 rows
    selectdo
        kasuh206.stat = kaprostat.cancel
        db.update(tkasuh206,db.retry,e)
        check.and.commit.or.abort.transaction()
    endselect

    db.retry.point()
    select  kasuh207.*
    from    kasuh207 for update
    where   kasuh207.ordid = {:kasuh206.ordid}
    as set with 1 rows
    selectdo
        invid = kasuh207.invid
    endselect


    db.retry.point()
    select  kasuh208.*
    from    kasuh208 for update
    where   kasuh208.invid = {:invid}
    as set with 1 rows
    selectdo
        kasuh208.stat = kaprostatpay.ordercancel
        db.update(tkasuh208,db.retry,e)
        check.and.commit.or.abort.transaction()
    endselect
}
|*--------------------------------------------------------------------------------
function extern return_order()
{
    |* Check if order is delivered
    if kasuh206.stat <> kaprostat.delivered then
        message("Cannot return this order. Only delivered orders can be returned.")
```

```
          else
          |* Check if return record already exists for this order

                    select kasuh209.*
                    from kasuh209
                    where kasuh209.ordid = :kasuh206.ordid
                    as set with 1 rows
                    selectempty
                              insert.return.record()
                              update.first.free.number.for.return()
                              update.order.status.to.return_in_progress()
                              refresh.curr.occ()

                    selectdo
                              message("A return request already exists for this order.")

                    endselect
          endif
}

function insert.return.record()
{
          select kasuh210.*  |* Get Return first free number
          from kasuh210
          where kasuh210.orig = kaproorig.ret
          as set with 1 rows
          selectdo
                    kasuh209.retid = kasuh210.ival & edit$(kasuh210.ffnu, "999")
          endselect

          kasuh209.ordid = kasuh206.ordid
          kasuh209.rsts  = kaprorsts.in_prog
          kasuh209.rfds  = kaprorfds.not_refunded
          kasuh209.rdte  = 0
          kasuh209.pdte  = 0

          ret = db.insert(tkasuh209, db.retry)
          check.and.commit.or.abort.transaction()
}

function update.order.status.to.return_in_progress()
{
          db.retry.point()
          select kasuh206.*
          from kasuh206 for update
          where kasuh206._index1 = {:kasuh206.ordid}
          as set with 1 rows
```

```
        selectdo
                kasuh206.stat = kaprostat.ret_in_prog
                ret = db.update(tkasuh206, db.retry)
                check.and.commit.or.abort.transaction()
        endselect
}

function update.first.free.number.for.return()
{
        db.retry.point()
        select kasuh210.*
        from kasuh210 for update
        where kasuh210.orig = kaproorig.ret
        as set with 1 rows
        selectdo
                kasuh210.ffnu = kasuh210.ffnu + 1
                ret = db.update(tkasuh210, db.retry)
                check.and.commit.or.abort.transaction()
        endselect
}
```

7.9 Invoices Program Scripts:

```
|****************************************************************************
|* kasuh0207  0  VRC B61C a  prmh
|* Invoice - First Free Number Logic
|* Suhita Pentakota
|* 2025-06-18
|****************************************************************************

|***************************** Declaration Section **************************

declaration:

        table   tkasuh207               |* Invoice Table
        table   tkasuh210               |* First Free Number Master

        domainkapro10s                          g.invid
        long    ret

|***************************** Program Section ******************************

before.program:
        set.synchronized.dialog("kasuh2107m000")

before.display.object:
        if fattr.occurnr = 1 then
                disable.fields("kasuh207.invid")
        endif

before.new.object:
        get.first.free.invoice.id()

after.update.db.commit:
        update.first.free.number.for.invoice()

|***************************** Field Functions Section ***********************

functions:

function extern payments()
{
        start.synchronized.child("kasuh2108m000",
                                 "kasuh207.invid",
                                 "kasuh208.invid")
}


function get.first.free.invoice.id()
{
        select  kasuh210.*
        from    kasuh210
        where   kasuh210.orig = kaproorig.inv
        as set with 1 rows
        selectdo
                kasuh207.invid = kasuh210.ival & edit$(kasuh210.ffnu, "999")
        endselect
}

function update.first.free.number.for.invoice()
{
        db.retry.point()
        select  kasuh210.*
        from    kasuh210 for update
        where   kasuh210.orig = kaproorig.inv
        as set with 1 rows
        selectdo
                kasuh210.ffnu = kasuh210.ffnu + 1
                ret = db.update(tkasuh210, db.retry)
                check.and.commit.or.abort.transaction()
        endselect
}

function check.and.commit.or.abort.transaction()
{
        if ret = 0 then
```

```
                commit.transaction()
        else
                abort.transaction()
        endif
}
```

7.10 Payments Program Scripts:

```
|******************************************************************************
|* kasuh0208  0  VRC B61C a  prmh
|* Payments - First Free Number Logic
|* Suhita Pentakota
|* 2025-06-18
|******************************************************************************

|****************************** Declaration Section ***************************

declaration:

        table   tkasuh208               |* Payment Table
        table   tkasuh210               |* First Free Number Master

        domain          kapro10s                        g.payid
        long    ret

|****************************** Program Section *******************************

before.program:
        set.synchronized.dialog("kasuh2108m000")

before.display.object:
        if fattr.occurnr = 1 then
                disable.fields("kasuh208.payid")
        endif

before.new.object:
        get.first.free.payment.id()

after.update.db.commit:
        update.first.free.number.for.payment()

|****************************** Field Functions Section ************************

functions:

function get.first.free.payment.id()
{
        select  kasuh210.*
        from    kasuh210
        where   kasuh210.orig = kaproorig.pay
        as set with 1 rows
        selectdo
                kasuh208.payid = kasuh210.ival & edit$(kasuh210.ffnu, "999")
        endselect
}

function update.first.free.number.for.payment()
{
        db.retry.point()
        select  kasuh210.*
        from    kasuh210 for update
        where   kasuh210.orig = kaproorig.pay
        as set with 1 rows
        selectdo
                kasuh210.ffnu = kasuh210.ffnu + 1
                ret = db.update(tkasuh210, db.retry)
                check.and.commit.or.abort.transaction()
        endselect
}

function extern paid(){

        db.retry.point()
        select  kasuh208.*
        from    kasuh208 for update
        where   kasuh208._index1 = {:kasuh208.payid}
        as set with 1 rows
        selectdo
                if kasuh208.stat = kaprostatpay.pending then
                        kasuh208.stat = kaprostatpay.paid
                        kasuh208.pydt = utc.num()
                        db.update(tkasuh208,db.retry,e)
                        check.and.commit.or.abort.transaction()
                else
                        message("Payment already completed or payment not applicable to order.")
                endif
        endselect

}



function check.and.commit.or.abort.transaction()
{
        if ret = 0 then
                commit.transaction()
        else
                abort.transaction()
        endif
}
```

7.11 Returns Program Script:

```
|******************************************************************************
|* kasuh0209  0  VRC B61C a  prmh
|* Returns - First Free Number Logic
|* Suhita Pentakota
|* 2025-06-18
|******************************************************************************

|****************************** Declaration Section ***************************

declaration:

        table   tkasuh206
        table   tkasuh207
        table    tkasuh208
        table    tkasuh209              |* Return Table
        table    tkasuh210              |* First Free Number Master

        domain          kapro10s                          g.retid
        long     ret

extern domain kapro10s invid
extern domain kapro10s ordid
extern domain kapro10s ordid1
|****************************** Program Section *******************************

before.program:
        set.synchronized.dialog("kasuh2109m000")

before.display.object:
        if fattr.occurnr = 1 then
                disable.fields("kasuh209.retid")
        endif

before.new.object:
        get.first.free.return.id()

after.update.db.commit:
        update.first.free.number.for.return()

|****************************** Field Functions Section ************************

functions:

function get.first.free.return.id()
{
        select  kasuh210.*
        from    kasuh210
        where  kasuh210.orig = kaproorig.ret
        as set with 1 rows
        selectdo
                kasuh209.retid = kasuh210.ival & edit$(kasuh210.ffnu, "999")
        endselect
}

function update.first.free.number.for.return()
{
        db.retry.point()
        select  kasuh210.*
        from    kasuh210 for update
        where   kasuh210.orig = kaproorig.ret
        as set with 1 rows
        selectdo
                kasuh210.ffnu = kasuh210.ffnu + 1
                ret = db.update(tkasuh210, db.retry)
                check.and.commit.or.abort.transaction()
        endselect
}

function check.and.commit.or.abort.transaction()
{
        if ret = 0 then
                commit.transaction()
        else
                abort.transaction()
        endif
}


|*-------------------------------------------------------------------------------------------

function extern confirm_return()
{
        select kasuh209.*
        from kasuh209 for update
        where kasuh209._index1 = :kasuh209.retid
        as set with 1 rows
        selectdo
                if kasuh209.rsts = kaprorsts.in_prog then
                        update.order.status.to.returned()
                        update.payment.status.to.refund_in_progress()
                        update.return.status.to.returned()
                        refresh.curr.occ()
                else
                        message("Return can be confirmed only if the return status is In Progress")
                endif
        endselect
}

function update.order.status.to.returned()
{
        db.retry.point()
        select kasuh206.*
        from kasuh206 for update
        where kasuh206._index1 = {:kasuh209.ordid}
        as set with 1 rows
        selectdo
                kasuh206.stat = kaprostat.returned
                ret = db.update(tkasuh206, db.retry)
                check.and.commit.or.abort.transaction()
        endselect
}

function update.payment.status.to.refund_in_progress()
{
        db.retry.point()
        select kasuh208.*
        from kasuh208 for update
        where kasuh208.invid in (
                select kasuh207.invid
                from kasuh207
                where kasuh207.ordid = :kasuh209.ordid
        )
        as set with 1 rows
        selectdo
                kasuh208.stat = kaprostatpay.ref_prog
                ret = db.update(tkasuh208, db.retry)
                check.and.commit.or.abort.transaction()
        endselect
}

function update.return.status.to.returned()
{
        db.retry.point()
        select kasuh209.*
        from kasuh209 for update
        where kasuh209._index1 = {:kasuh209.retid}
        as set with 1 rows
        selectdo
                kasuh209.rsts = kaprorsts.returned
                kasuh209.pdte = utc.num()  |* Set return date to current UTC timestamp
                ret = db.update(tkasuh209, db.retry)
                check.and.commit.or.abort.transaction()
        endselect
}

|*-------------------------------------------------------------------------------------

function extern confirm_refund()
{
        select kasuh209.*
        from kasuh209
        where kasuh209._index1 = {:kasuh209.retid}
        as set with 1 rows
        selectdo
                if kasuh209.rsts = kaprorsts.returned and kasuh209.rfds = kaprorfds.not_refunded then
                update.return.refund.status.to.refunded()
                update.order.status.to.returned.and.refunded()
                update.payment.status.to.refunded()
                refresh.curr.occ()
            else
                message("Return not completed/Refunded already completed")
            endif
        endselect

}

function update.return.refund.status.to.refunded()
{
        db.retry.point()
        select kasuh209.*
        from kasuh209 for update
        where kasuh209._index1 = {:kasuh209.retid}
        as set with 1 rows
        selectdo
            kasuh209.rfds = kaprorfds.refunded
            kasuh209.rdte = utc.num()
            ret = db.update(tkasuh209, db.retry)
            check.and.commit.or.abort.transaction()
        endselect
}

function update.order.status.to.returned.and.refunded()
{
        db.retry.point()
        select kasuh206.*
        from kasuh206 for update
        where kasuh206._index1 = {:kasuh209.ordid}
        as set with 1 rows
        selectdo
            kasuh206.stat = kaprostat.ret_ref
            ret = db.update(tkasuh206, db.retry)
            check.and.commit.or.abort.transaction()
        endselect
}

function update.payment.status.to.refunded()
{
        db.retry.point()
        select kasuh208.*
        from kasuh208 for update
        where kasuh208.invid in (
            select kasuh207.invid
            from kasuh207
            where kasuh207.ordid = :kasuh209.ordid
        )
        as set with 1 rows
        selectdo
            kasuh208.stat = kaprostatpay.refunded
```

```
                ret = db.update(tkasuh208, db.retry)
                check.and.commit.or.abort.transaction()
        endselect
}
```

7.12 First Free Number Program Script:

```
|*******************************************************************************
|* kasuh2110  0  VRC B61C a  prmh
|* First Free Number
|* Joshna
|* 2025-06-16
|*******************************************************************************
|* Main table kasuh210 First Free Number, Form Type 1
|*******************************************************************************

|****************************** declaration section ***************************
declaration:

  table   tkasuh210 | First Free Number


|****************************** program section ********************************


|****************************** group section **********************************
```

References:

Material :

- Programmers Guide 10.4.1

Website:

- https://docs.infor.com/ln/10.4/en-us/lnolh/docs/ln_10.4_devtoolsdg__en-us.pdf

- https://shreeragula.github.io/Procurement-Sales-Management-System/