Ohioedge CRM Administrator's Reference Guide

Covers Organization, Customer, Data Publisher, and Workflow Components

By Tammy Dixit

April 5, 2003

Table of Contents

Overview

Ohioedge CRM is a Web-based, Enterprise Java (J2EE) Customer Management Application specifically designed for $5-200M organizations requiring centralized, enterprise-wide, co-ordination of sales generation and sales fulfillment activities. At the core of Ohioedge CRM is a workflow management, campaign management, customer management, contact management, and template-driven data publishing functionality. Built upon Ohioedge J2eeBuilder, Ohioedge CRM can be easily extended with additional functionality, as well as plugged-into an existing ERP system.

Ohioedge CRM is composed of Ohioedge Organization, Ohioedge Customer, Ohioedge Publisher, and Ohioedge Workflow components. Configuring Ohioedge CRM consists of configuring these components it is made up of. This guide provides a thorough walk-through of configuring Organization, Customer, Publisher and Workflow components.

"Link->Link->…" is the nomenclature used to represent the path or how-to get to the Web page of a component. A "business component" is a generic reference to well-defined business functionality. For example Employee business component is a reference to organizational employee as a functional entity. Configuring a business component in the context of this document involves creating, updating, deleting or selecting of its data.

Business components requiring setup are registered with the Setup component and are accessible from @Quick Links->Setup. Figure 1 shows the Setup component's home page.

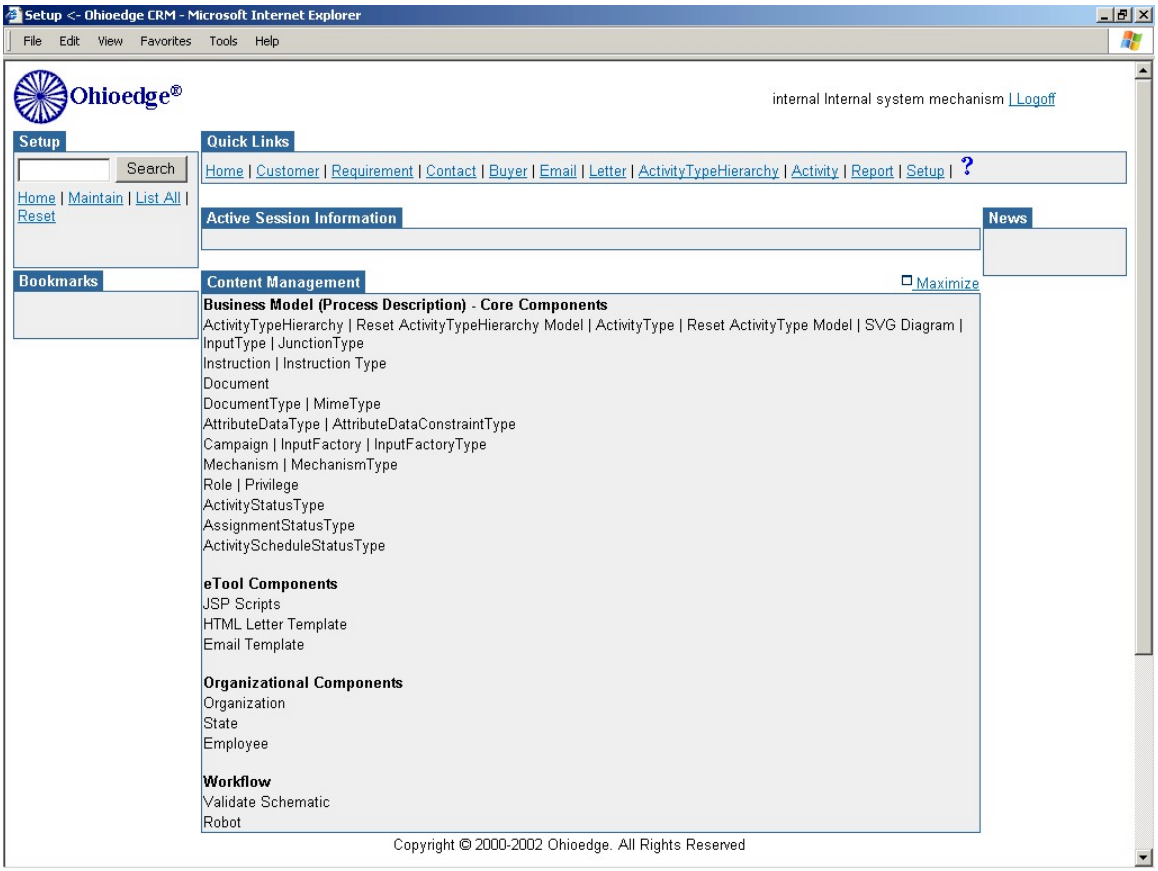


Figure 1. Setup Menu

Ohioedge Workflow Component Setup

Ohioedge Workflow is a 100% Web-based, generic workflow management engine. The generic behavior of Ohioedge Workflow is achieved by storing all of its specific properties in the underlying database. There is nothing "hard-coded" in the source-code; making Ohioedge Workflow a data-driven, generic J2EE component. Its generic-ness makes it configurable for different scenarios, such as Customer Relationship Management, Claims Processing, Manufacturing Process Control, Billing, etc. In the context of this guide, we will be configuring Ohioedge Workflow to model a Customer Relationship Management (CRM) scenario. This activity involves constructing of CRM-specific data in the underlying data structure, a process we refer to as "Setup."

The order in which the business components are setup is as below:

1. Setup independent value lists or the basic components that are not dependent on other business components. For example, the Name Prefix business component.

2. Setup dependent business components. For example the Employee business component, as it is dependent on the Name Prefix business component.

1. ActivityType

Location: @Setup->ActivityType->Maintain or ListAll-><record to maintain>

Dependency: None.

The ActivityType business component is at the core of Ohioedge Workflow. It represents the types of activities performed within an organization. In the context of Ohioedge Workflow, there are two categories of activity types, 1) UOB activity type, and 2) Scenario (non-UOB) activity type. An UOB activity type is the smallest unit of behavior that can not be further decomposed into smaller activity types. Transactions can only be recorded at UOB activity types. Also work items (see InputType) can only flow to and from UOB activity types. A scenario activity type is a summary-level activity type. It is always a parent of either another scenario activity type(s) or UOB activity types (see ActivityTypeHierarchy.)

Example:

Name	Category	Description
ColdCall	UOB	Making cold calls to customer
PrntLtr	UOB	Printing literature to be mailed to customer
FollowCall	UOB	Following up with customer
SalesCall	UOB	Meeting with customer
OrdComp	UOB	Order complete
NI	UOB	Not Interested
EmailLtr	UOB	Emailing literature to customer
LtrReq	UOB	Literature requested by customer
RcvOrd	UOB	Order received from customer
EngrRevw	UOB	Reviewing procedures by the engineering dept.
Acctng	UOB	Accounting department procedures
Prod	UOB	Production department procedures
Shpng	UOB	Shipping procedures
DemoOrg	SCENARIO	Demonstration organization
US Sales	SCENARIO	US Sales
DH	SCENARIO	DH-Central Region
GP	SCENARIO	GP-Western Region
TG	SCENARIO	TG-Central Region
CF	SCENARIO	CF-Florida Region

In the above example, ColdCall is an UOB activity type. It is a performable activity type. For example, a Sales Rep (or the person assigned to this task/activity type) would make a cold call and record the details of the activity. In case of EmailLtr, the assignee would email the literature

requested by a customer and record the transaction. US Sales is a child of DemoOrg. Both DemoOrg and US Sales are scenario activity types. There are no transactions occurring at these activity types.

A **vertical model**, typically an organizational chart, is built using scenario activity types. A **horizontal model**, typically a business process model or a workflow, is built using UOB activity types. Figure 2 illustrates a workflow (horizontal model) representing relationships between the UOB activity types listed in the example above.

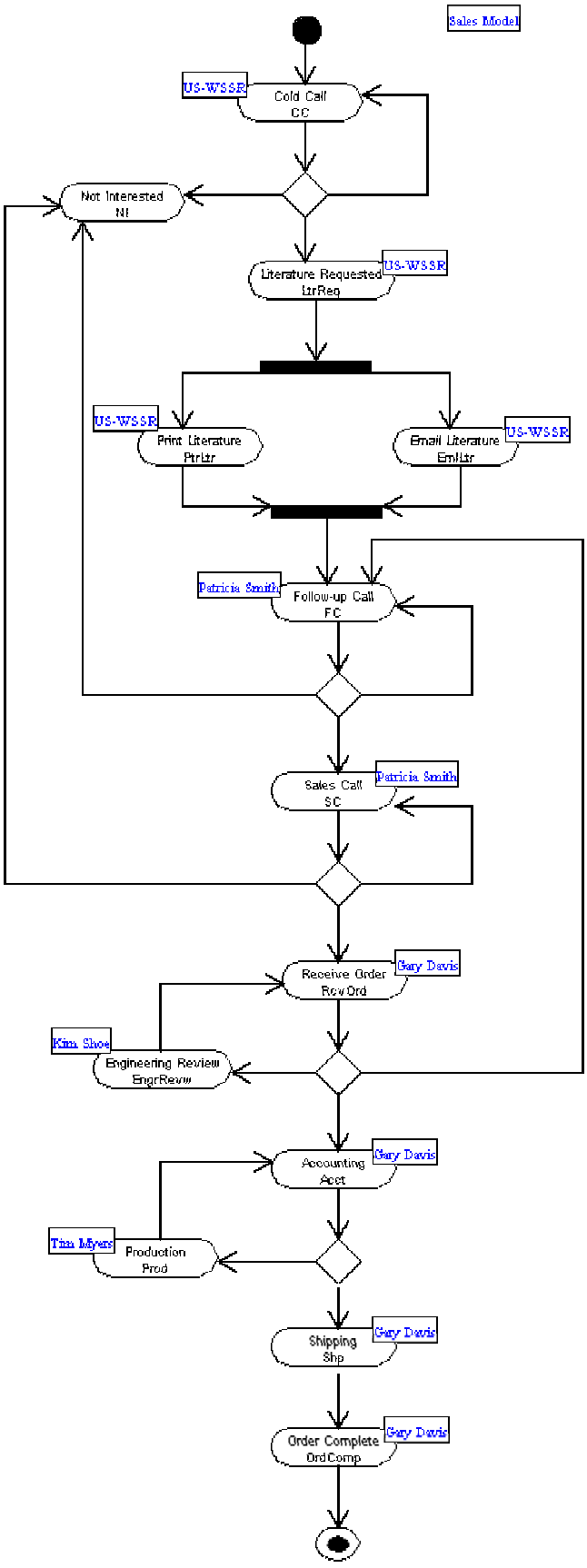


Figure 2.

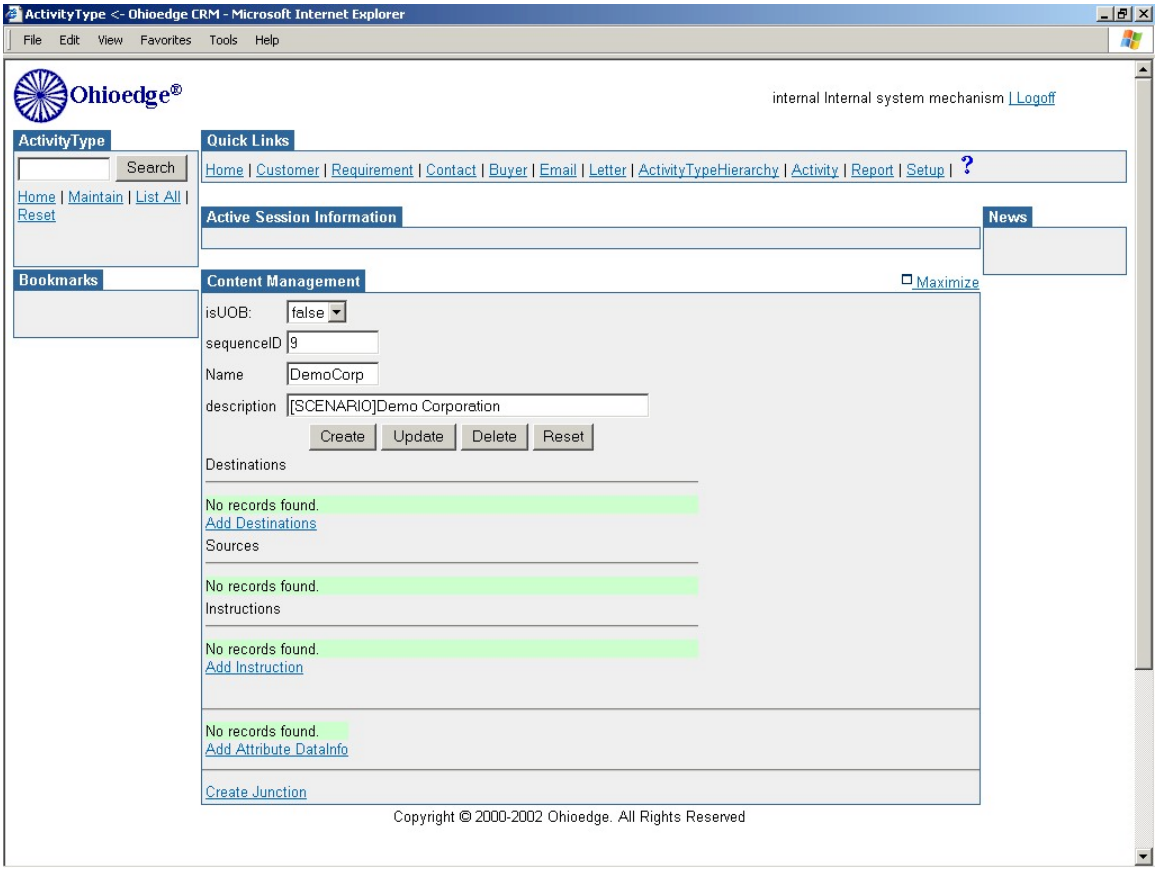


Figure 3.

Figure 3 shows the ActivityType component's maintenance page.

2. JunctionType:

Location: @Setup->JunctionType->Maintain or ListAll-><record to maintain>

Dependency: None.

The JunctionType business component represents the types of junctions available for modeling splits and merges of a workflow. In the context of Ohioedge Workflow, a workflow is a graph of inter-related activity types. It represents a flow of work (see InputType) between UOB activity types.

Example:

Name	Description
AND	AND junction type
OR	OR junction type
XOR	XOR junction type

As illustrated in Figure 2, when a cold call is made there are three possible outcomes:

1) The person is not found **or**

2) The person is not interested **or**

3) The person requests literature

Here, out of three, **only one** outcome is possible. Thus it is an XOR (Exclusive OR) junction.

When the literature is requested there are two paths that are simultaneously followed:

1) Printing the literature to be mailed to the customer **and**

2) Emailing the literature to the customer

Here, **all** the outcomes are simultaneously followed. Thus it is an AND junction.

An OR junction is where either one or more than one outcome is possible.

3. DataType

Location: @Setup->DataType->Maintain or ListAll-><record to maintain>

Dependency: None.

The DataType business component represents the type of data associated with a custom field. Each activity type is unique in nature. In Ohioedge Workflow, custom fields can be created and associated with an activity type for capturing its unique data.

Example:

Name	Description	JavaType	SqlType
Integer	Integer data type	java.lang.Integer	X
String	String data type	java.lang.String	X
Timestamp	Timestamp data type	java.sql.Timestamp	X

4. DataConstraint

Location: @Setup->DataConstraint->Maintain or ListAll-><record to maintain>

Dependency: None.

The DataConstraint business component defines the data integrity constraint imposed by a custom field. Just to give an example, let's say we need to capture the duration of the phone call

information during the ColdCall activity type. To capture the data we create and assign an attribute – 'phone call duration' to the ColdCall activity type. By imposing the constraint of NOTNULL on the 'phone call duration' custom field would *force* the activity performer to enter the duration data.

Example:

Name	**Description**
NOTNULL	NULL/empty value is not allowed
NULL	NULL/empty value is allowed

5. ActivityTypeHierarchy

Location: @Setup->ActivityTypeHierarchy->Maintain or ListAll-><record to maintain>

Dependency: ActivityType

The ActivityTypeHierarchy business component represents the hierarchical (vertical) relationship between a parent activity type (always a scenario activity type) and a child activity type (either an UOB or a scenario activity type.) A root level activity type hierarchy has a NULL parent activity type. The current release of Ohioedge CRM allows only one root-level activity type hierarchy. If you create more than one root-level activity type hierarchies, you will need to delete all extra root level activity type hierarchies by directly executing 'delete from ActivityTypeHierarchy where "activityTypeHierarchyID" = ?' SQL statement at the database level.

Example:

Figure 4 shows the hierarchical model of Demonstration organization (DemoOrg).

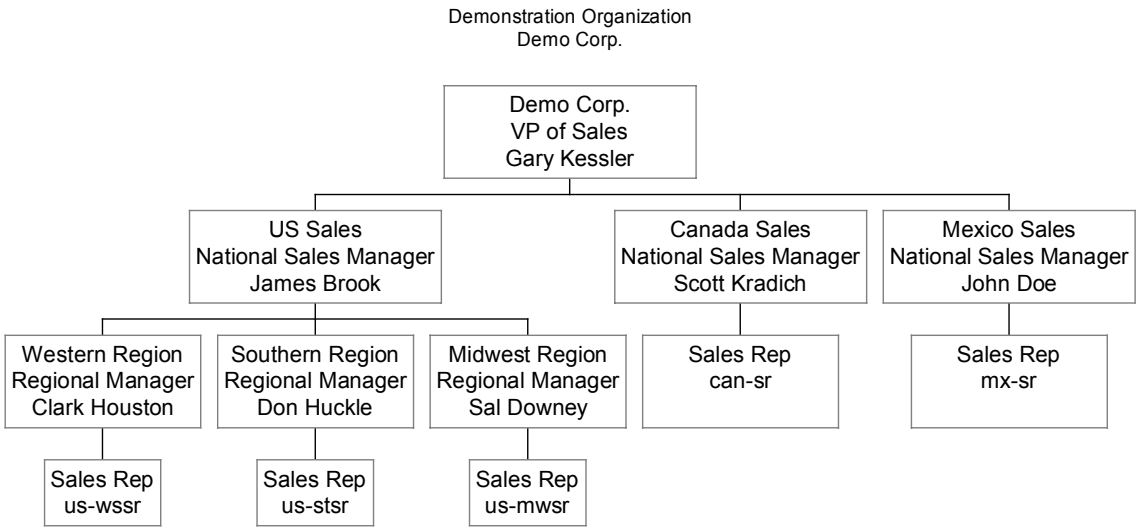


Figure 4. Demo Corp. Organizational Model

Demo Corp operates in three countries and has one national sales manager for every country. Under US national office, it has three regions. Each region has a designated Regional Manager overseeing the activities for the region. Sales Reps work for their region. In case of Canada and Mexico national offices, there are no regional offices. Sales Reps directly work for their national offices. The organizational model defines organizational hierarchy, people hierarchy, and roles and responsibilities associated with the levels of hierarchy.

Figure 5 below shows the ActivityTypeHierarchy component's data maintenance page.

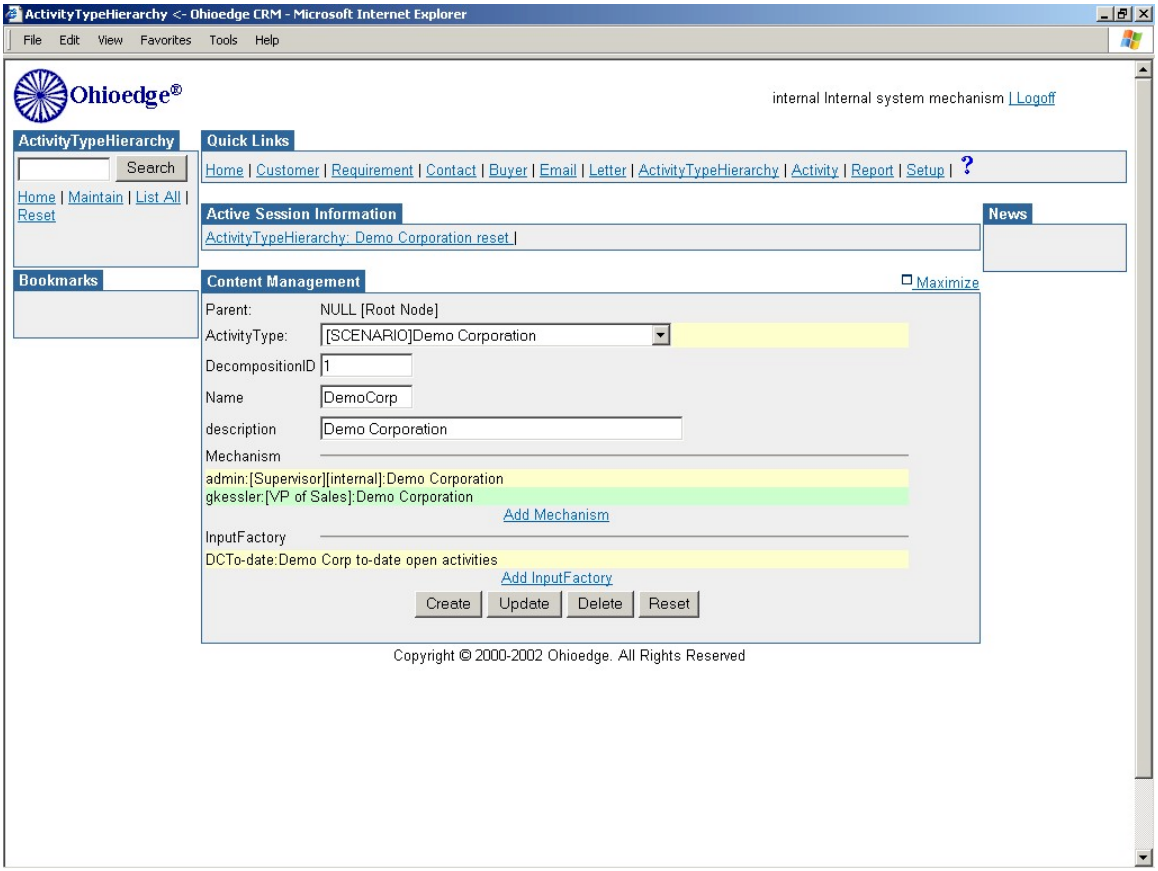


Figure 5.

6. MechanismType

Location: @Setup->MechanismType->Maintain or ListAll-><record to maintain>

Dependency: None.

The *MechanismType* business component specifies the types of resources.

Example:

Name	Description
Email	Resource of the type email
Emp	Resource of the type employee

7. Mechanism

Location: @Setup->Mechanism->Maintain or ListAll-><record to maintain>

Dependency: MechanismType.

The *Mechanism* business component specifies workflow resources.

Example:

Name	Type	Description
John Smith	Emp	John Smith
Pat Doe	Emp	Patricia Doe
Dave Burns	Emp	David Burns

8. Role

Location: @Setup->Role->Maintain or ListAll-><record to maintain>

Dependency: None.

The *Role* business component specifies workflow security roles.

Example:

Name	Description
Supervisor	User with a Supervisor role
Manager	User with a Manager role
Worker	User with a Worker role

9. Privilege

Location: @Setup->Privilege->Maintain or ListAll-><record to maintain>

Dependency: None.

The *Privilege* business component specifies workflow security privileges.

Example:

Name	Description
Approve	Privilege to approve an assignment
Sign-off	Privilege to sign-off an assignment
Approve	Privilege to approve an activity
OnHold	Privilege to put activity schedule on hold
Release	Privilege to release activity schedule
Route	Privilege to route activity schedule
Assign	Privilege to assign assignments
Schedule	Privilege to schedule activities
Originator	Privilege to originate or create activities

10. RolePrivilege

Location: @Setup->Role->Maintain or ListAll-><record to maintain>->Add Privilege or <record to maintain>

Dependency: Role, Privilege.

The *RolePrivilege* business component specifies the relationship between roles and privileges.

Example:

Role	Privilege
Supervisor	Assign: Supervisor can assign Assignments to other Mechanisms Approve: Supervisor can approve an Activity or Assignment signed-off by other Mechanisms Sign-off: Supervisor can sign-off an Assignment Originator: Supervisor can create or originate an Activity
Manager	Schedule: Manager can schedule an Activity to be sent to the next ActivityType Route: Manager can route an Activity to the next ActivityType OnHold: Manager can schedule an Activity to be put on-hold until, certain business conditions are met to release the Activity Release: Manager can schedule an Activity to be released after all the business conditions for that Activity are met
Worker	Sign-Off: Worker can sign-off an Assignment

11. ActivityTypeHierarchyMechanism

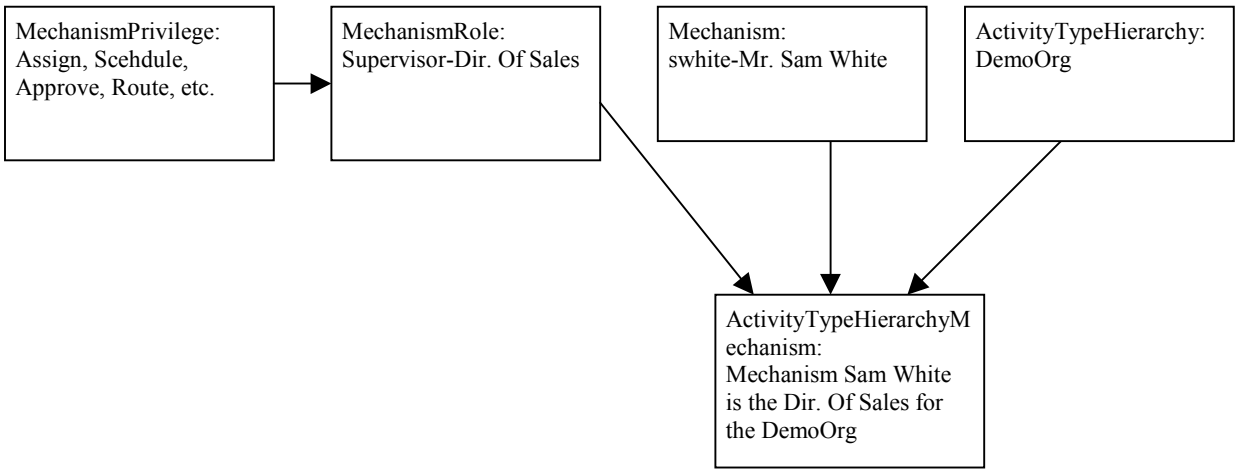
Location: @Setup->ActivityTypeHierarchy->Maintain or [ListAll-><record to maintain>]->Add Mechanism or <record to maintain>

Dependency: ActivityTypeHierarchy, Mechanism.

The *ActivityTypeHierarchyMechanism* business component specifies the relationship between activity type hierarchies and mechanisms.

Example:

Figure 6 below shows how a mechanism is attached to an activity type hierarchy.



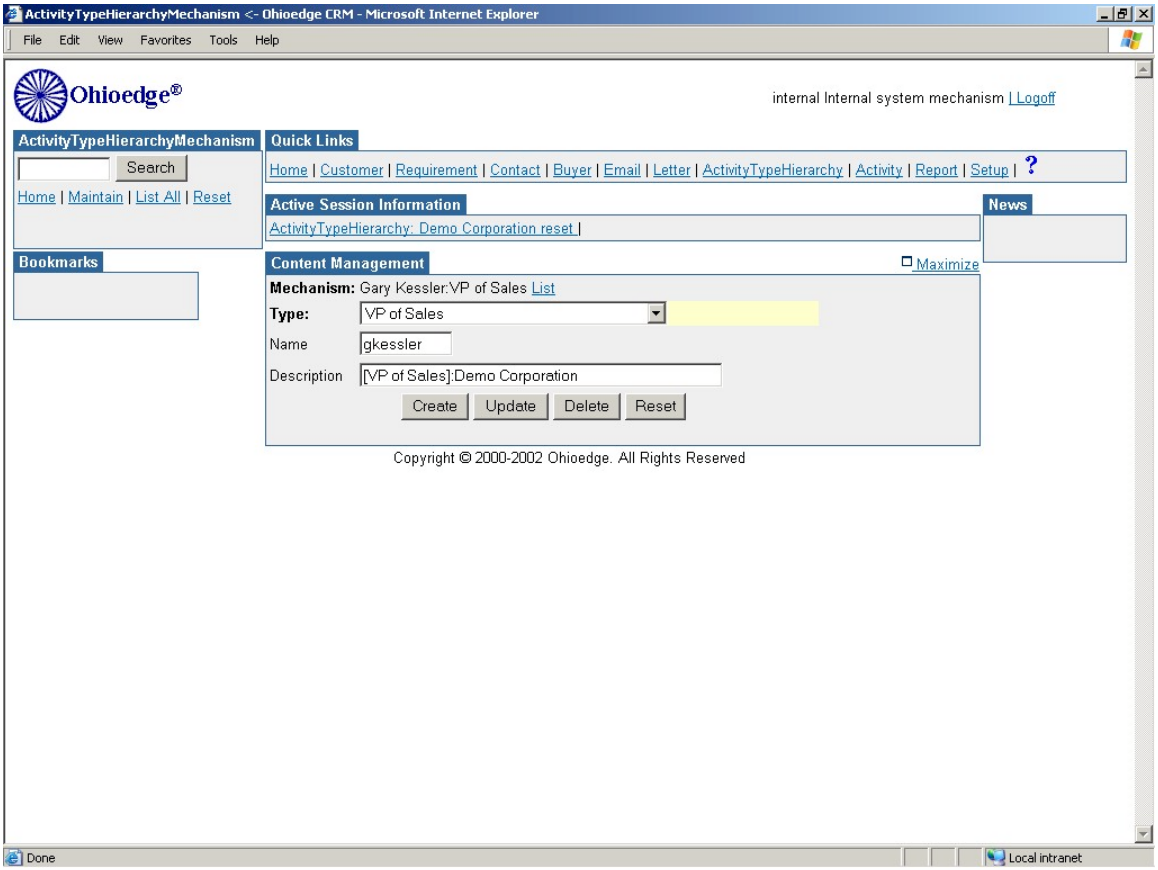


Figure 6.

The table below shows how different mechanisms inherit different roles at different activity types in Demo Corp organizational model.

	Mechanism	**MechanismRole**	**ActivityTypeHierarchy**	**Username/pwd**
1.	Sam White	Dir. Of Sales	DemoOrg Scenario	guest7/guest7
2.	David Burns	Regional Manager	DH–Central Scenario	guest4/guest4
	Jeff Todak	Sales Rep	DH-Central Scenario	guest8/guest8
3.	Tim Myers	Regional Manager	DB-Midwest Scenario	guest5/guest5
	Kim Wentley	Sales Rep	DB-Midwest Scenario	guest9/guest9
5.	Gary Davis	Regional Manager	TG-Southern Scenario	guest6/guest6
	Mark Sears	Sales Rep	TG-Southern Scenario	guest10/guest10
6.	David Burns	Regional Manager	GP-Western Scenario	guest4/guest4
7.	John Doe	Regional Manager	CF-Florida	guest/guest
8.	Jeff Hatt	Worker	All DH ActivityTypes	guest5/guest5
9.	Kim Wentley	Worker	All DB ActivityTypes	guest9/guest9
10.	Mark Sears	Worker	All TG ActivityTypes	guest10/guest10

12. Activity

Location: @QuickLinks->Activity->Maintain or ListAll-><record to maintain>

Dependency: ActivityType, InputType, ActivityTypeHierarchy, ActivityTypeHierarchyMechanism.

The *Activity* business component specifies the transactions performed by mechanisms at UOB activity types. **It is not a setup component.** It is included in this guide only for the clarity of the overall workflow architecture.

Example:

The ColdCall activity type implements a business rule – 'Call an organization and validate it as a prospective client.' The mechanism "John Smith" performs a business transaction by "making a cold call", as required by the "Cold Call" activity type. This transaction is an activity.

Figure 7 shows the Activity component's maintenance page.

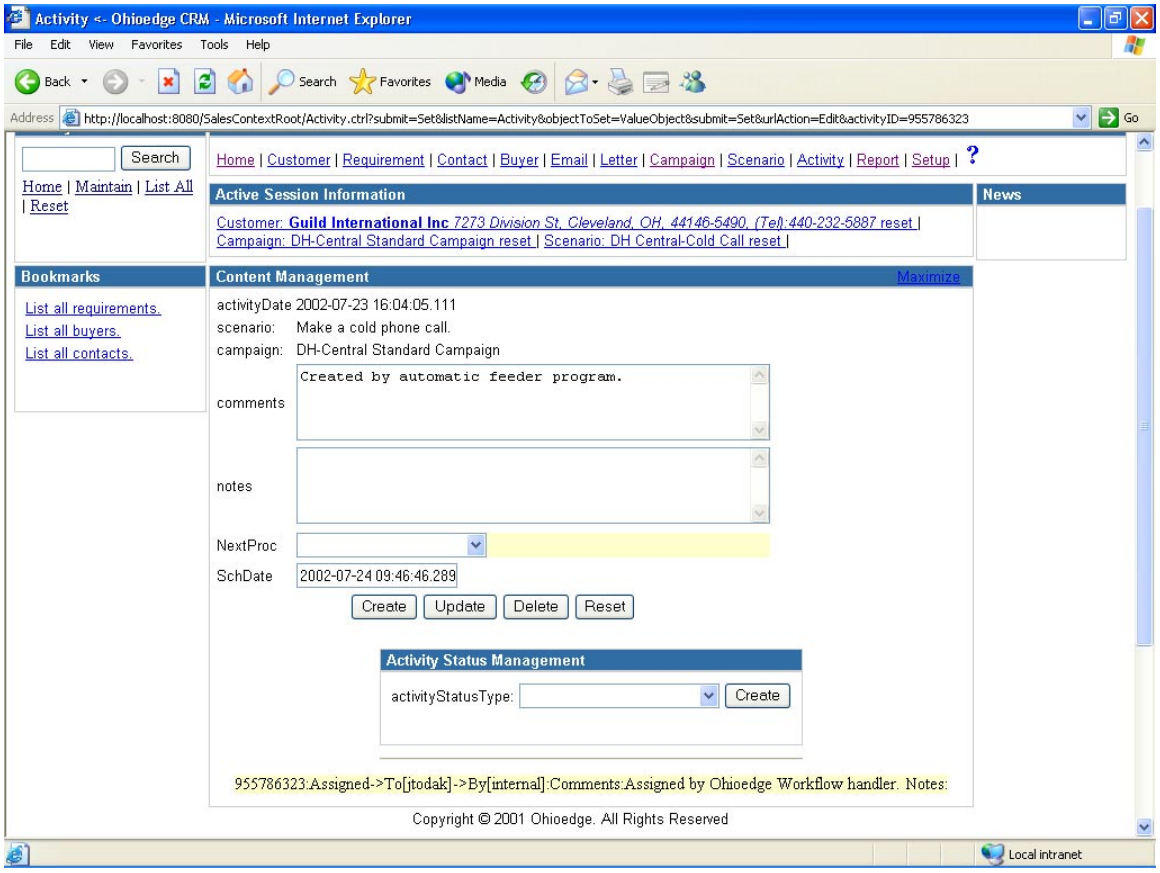


Figure 7.

13. ActivityStatusType

Location: @Setup->ActivityStatusType->Maintain or ListAll-><record to maintain>

Dependency: None.

The *ActivityStatusType* business component specifies the types of activity statuses. Every activity status has a level associated with it indicating its position among all activity status type. The example below illustrates the significance of a level.

Example:

Once an activity has "Approve" status it cannot be changed to "Signed-off" because "Approve" is at a higher level than "Sign-off."

Name	Level	Description
Sign-off	10	Activity Sign-Off Privilege. Mechanism performs an Activity. After satisfying the business rule the Mechanism signs-off the activity indicating that the Activity has been completed and is either waiting to be approved or released by a Mechanism with appropriate privileges.
Approve	20	Activity Approve Privilege. An Activity can be approved by a Mechanism based on the business rules for that ActivityType

14. ActivityStatusTypePrivilege

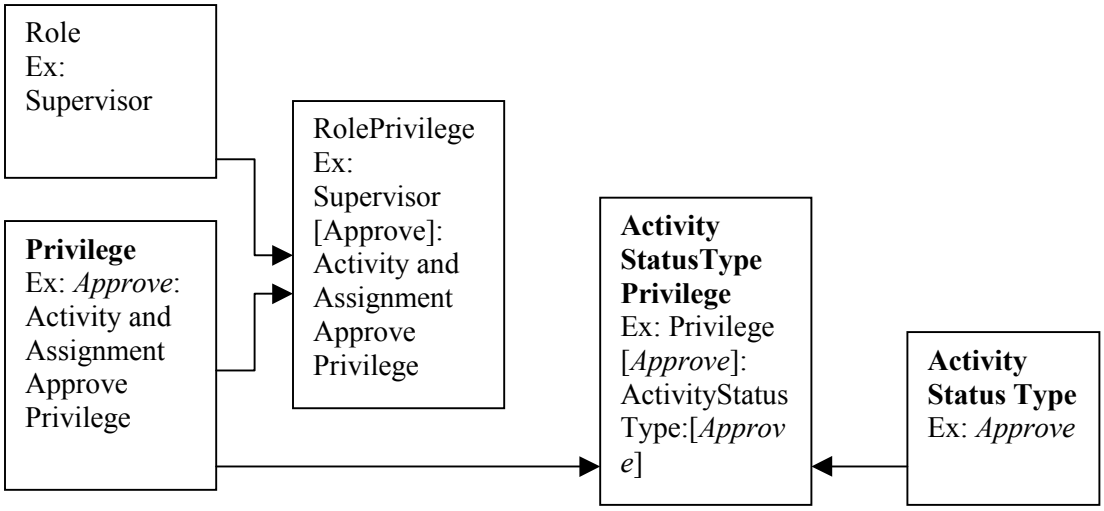
Location: @Setup->ActivityStatusType->[Maintain or ListAll-><record to maintain>]->Add Privilege or <record to maintain>

Dependency: ActivityStatusType, Privilege.

The *ActivityStatusTypePrivilege* business component represents the relationship between activity status types and privileges.

Example:

The diagram below shows the "Approve" privilege is associated with the "Approve" activity status type.



15. AssignmentStatusType

Location: @Setup->AssignmentStatusType->Maintain or ListAll-><record to maintain>

Dependency: None.

The *AssignmentStatusType* business component specifies the types of assignment statuses. An assignment is the association between a mechanism and a task. Every assignment status type

has a level associated with it indicating its position among all assignment status types. The example below illustrates the significance of a level.

Example:

A mechanism with Originate privilege creates or originates an activity 'Make a cold call.' The assignment logic of the workflow then assigns the activity to a mechanism that has Sign-off privilege. The assigned mechanism then in-turn performs the task and signs-off indicating that the task is completed and the assignment can be approved or activity can be released.

Name	Level	Description
Sign-Off	10	Assignment Sign-Off Privilege. Mechanism with Assign privilege then assigns the task of performing the assignment to another Mechanism. This Mechanism then in-turn performs the task and signs-off the assignment indicating that the task is completed.
Approve	20	Assignment Approve Privilege. Assignment that is signed-off can be approved by a Mechanism with appropriate privilege based on the business rules for that ActivityType

16. AssignmentStatusTypePrivilege

Location: @Setup->AssignmentStatusType->[Maintain or ListAll-><record to maintain>]->Add Privilege or <record to maintain>

Dependency: AssignmentStatusType, Privilege.

The *AssignmentStatusTypePrivilege* business component represents the relationship between assignment status types and privileges.

17. ActivityScheduleStatusType

Location: @Setup->ActivityScheduleStatusType->Maintain or ListAll-><record to maintain>

Dependency: None.

The *ActivityScheduleStatusType* business component represents the types of activity scheduel statuses. Every activity schedule status type has a level associated with it indicating its position among all activity schedule status types.

Example:

Name	Level	Description
OnHold	10	ActivitySchedule OnHold Privilege. ActivitySchedule OnHold Status does not let the Activity to be routed to the next ActivityType
Release	20	ActivitySchedule Release Privilege. After the Activity is completed it is ready to be released
Route	30	ActivitySchedule Route Privilege. ActivitySchedule Route Status is internally used by the Robot to mark an activityschedule as routed.

18. ActivityScheduleStatusTypePrivilege

Location: @Setup->ActivityScheduleStatusType->[Maintain or ListAll-><record to maintain>]->Add privilege or <record to maintain>

Dependency: ActivityScheduleStatusType, Privilege.

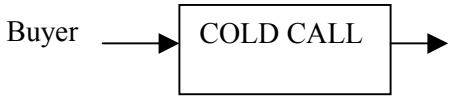
The *ActivityScheduletStatusTypePrivilege* business component represents the relationship between activity schedule status types and privileges

19. InputType

Location: @Setup->InputType->Maintain or ListAll-><record to maintain>

Dependency: None.

The *InputType* business component specifies the types of input that could flow through the workflow. For example,



Name	Description
Customer	Activity is peformed on customers. For example, a cold call was made to a customer - Rainbow Printings.
CustReq	Activity is peformed on customer requirements. For example, a cold call was made to a customer that has a specific requirement - IT services.
Buyer	Activity is performed on buyers. For example, a cold call was made to – Mr. Rob Smith, person in charge of buying IT services at Rainbow Printings.
CustPer	Activity is performed on customer persons. For example, a cold call was made to - Mr. Gary Richardson, a contact at Rainbow Printings.

20. InputFactoryType

Location: @Setup->InputFactoryType->Maintain or ListAll-><record to maintain>

Dependency: None.

The InputFactoryType business component represents the types of input factories.

Example:

Name	Description	Explanation
Report	Activity Report	Input factories used by ActivityTypes HTML reporting module
Loader	Input Loader	Input factories used by the Campaign management module

21. InputFactory

Location: @Setup->InputFactory->Maintain or ListAll-><record to maintain>

Dependency: InputFactoryType, InputType.

The *InputFactory* business component represents a mechanism for fetching inputs from internal or external data sources. In technical terms, it is a generic SQL query tool that allows selection of records from a data-source.

Example:

The R1 input factory is based on the SQL query that finds all the activities to date that do not have an activity schedule of 'Routed.'

FactoryType: Report

InputFactory Name: R1-to date

Description: Report 1 - To-date activities based on Activity Schedule

SQL Query:

```
SELECT "Activity".*, "ActivityStatus"."name" "activityStatusName"
FROM "Activity" LEFT OUTER JOIN "ActivityStatus" ON "Activity"."activityID" = "ActivityStatus"."activityID"
WHERE "activityTypeID" = ? AND NOT EXISTS (SELECT 'TRUE' FROM "ActivitySchedule", "Activation"
WHERE "ActivitySchedule"."activityID" = "Activation"."successorID" AND "Activation"."predecessorID" =
"Activity"."activityID") AND NOT EXISTS (SELECT "ActivityScheduleStatus"."activityScheduleID" FROM
"ActivitySchedule", "ActivityScheduleStatus"
WHERE "ActivityScheduleStatus"."activityScheduleID" = "ActivitySchedule"."activityScheduleID"
AND "ActivitySchedule"."activityID" = "Activity"."activityID" AND
"ActivityScheduleStatus"."activityScheduleStatusTypeID" IN (SELECT "activityScheduleStatusTypeID" FROM
"ActivityScheduleStatusType" WHERE "ActivityScheduleStatusType"."name" IN ('Route'))) ORDER BY
"activityDate" desc
```

22. ActivityTypeHierarchyInputFactory

Location: @Setup->[ActivityTypeHierarchy]->[Maintain or ListAll-><record to maintain>]->[Add InputFactory or <record to maintain>]

Dependency: ActivityTypeHierarchy, InputFactory.

The *ActivityTypeHierarchyInputFactory* business component represents the relationship between activity type hierarchies and input factories.

Example:

The Director Of Sales of Demo Corp needs to get a report of all the activities under the regions Central, Midwest, Southern, Western and Florida for a review. By assigning the input factory R1/Report 1 to the activity type hierarchy – Demo Corp would enable the Director of Sales (who is assigned as a mechanism to Demo Corp.) to view the result - all the open activities that fall under the umbrella of Demo Corp.

Figure 8 shows how an input factory is assigned to an activity type hierarchy.

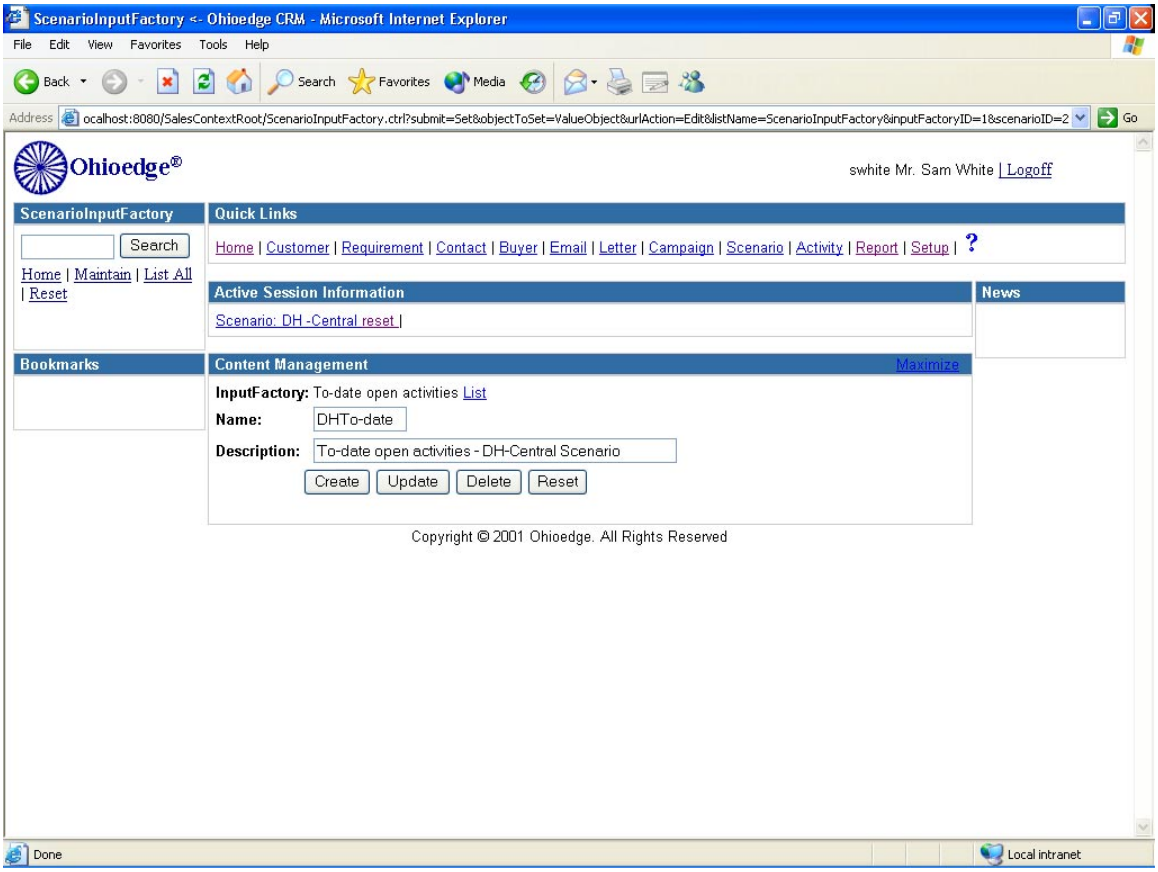


Figure 8.

23. Workflow Engine

The *workflow engine* business component primarily automates the following four activities:

Assign – The Assignor component manages automatic assignments of activities to resources based on custom assignment rules.

Perform – The Performer component automatically performs tasks on the behalf of assignee. For example, email performer automatically sends email messages to the recipients. Performer is a client-side (Web) component. Performer API can be easily extended for other types of tasks.

Schedule – The Scheduler component automatically schedules completed task based on custom scheduling rules. Custom scheduling rules are written as JavaBeans by extending Schedule Rules API.

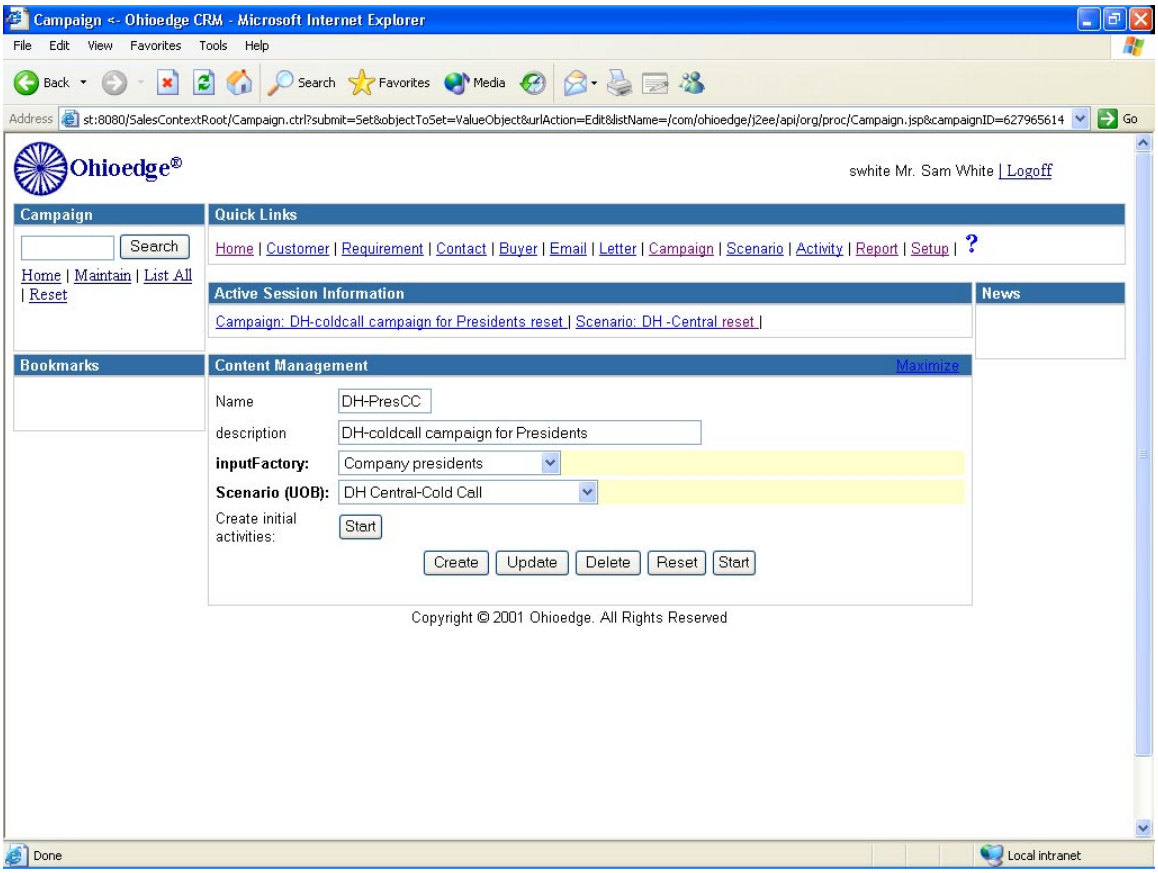
Route – The Router component manages automatic routing of activities to the next processes in the business process model.

The service manager component configures workflow engine execution parameters, such as, frequency of execution, duration of execution, etc.

24. Campaign Management

Location: @Setup->Campaign->Maintain or ListAll-><record to maintain>

The Campaign business component provides functionality to load a set of Inputs into the workflow model. All the activities made against the set of Inputs loaded by a campaign are associated with it throughout their life cycle. Figure 9 shows how campaigns are created.



Example:

DHPresCC is a campaign to call the presidents of companies in the DH-Central region. It uses 'Company presidents' input factory to fetch presidents and creates an initial activity at 'DH Central-Cold Call' activity type. Below is the SQL associated with the 'Company presidents' input factory:

```
SELECT cp.*
FROM    "Customer" c, "CustomerPerson" cp
WHERE  c."customerID" = cp. "customerID"
                        AND cp. "title" LIKE "President"
                        AND c. "areaCode" LIKE "216"
```

The j2ee-builder.xml <interface> tag provides the CustomerPerson->Activity mapping schema. The campaign component uses this schema to convert the fetched records into activities.

25. MimeType

Location: @Setup->MimeType->Maintain or ListAll-><record to maintain>

Dependency: None.

The *MimeType* business component specifies mime types.

Example:

TypeName	Description
Jsp	Text/jsp

26. InstructionType

Location: @Setup->InstructionType->Maintain or ListAll-><record to maintain>

Dependency: None.

The *InstructionType* business component specifies types of instructions.

Example:

TypeName	Description
Email	Send an e-mail
Print	Print a document

27. Instruction

Location: @Setup->Instruction->Maintain or ListAll-><record to maintain>

Dependency: InstructionType.

The *Instruction* business component specifies directives for executing activities.

Example:

- OeEmail: Send an Email (InstructionType) based on the document OeInfo.

- OePrint: Print (InstructionType) an envelope based on the document OeInfoEnv.

Figure 10 shows how to setup an instruction.

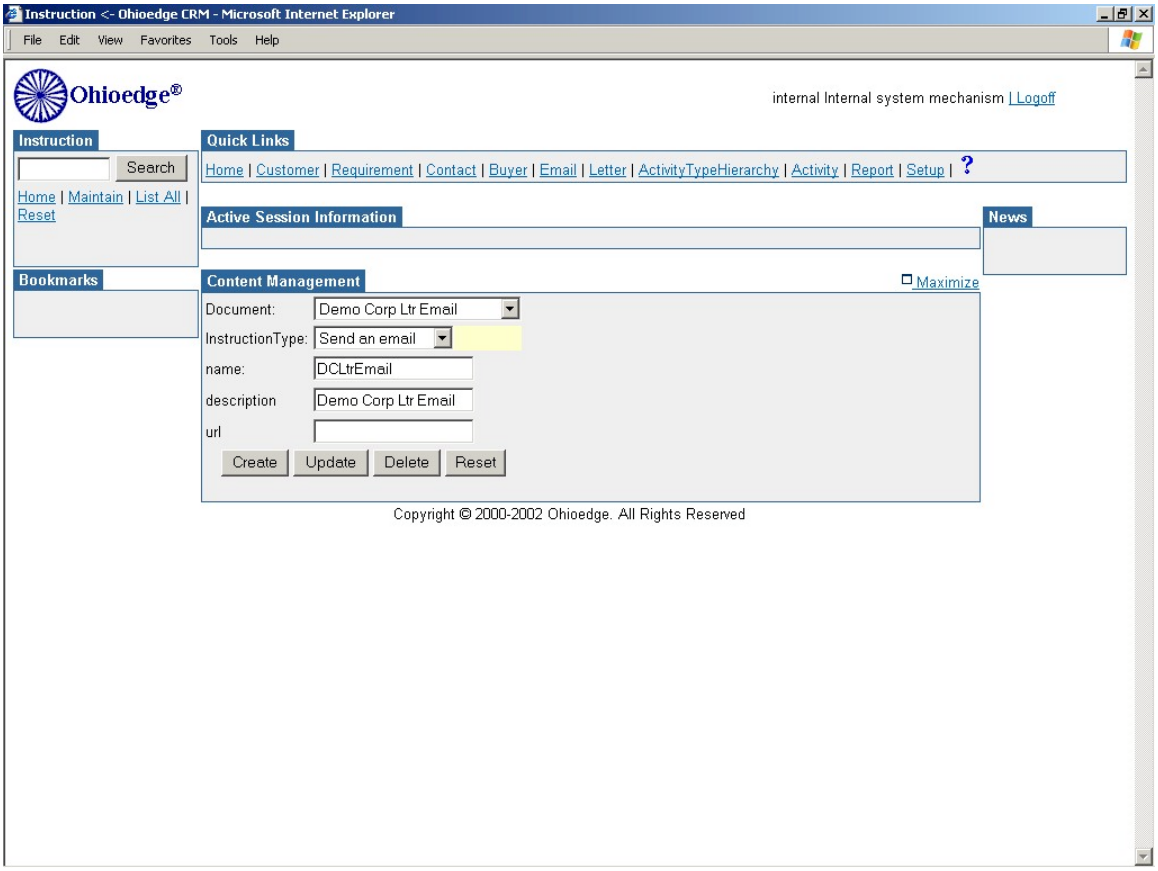


Figure 10.

28. DocumentType

Location: @Setup->DocumentType->Maintain or ListAll-><record to maintain>

Dependency: DocumentType.

The *DocumentType* business component specifies types of documents.

Example:

TypeName	Description
EmailTmpt	Email Template
LetterTmpt	Letter Template

29. Document

Location: @Setup->Document->Maintain or ListAll-><record to maintain>

Dependency: DocumentType.

The *Document* business component specifies internal or external documents. Currently internal documents recognized by the workflow are email and letter templates.

Example:

Figure 11 shows the Document business component's maintenance page.

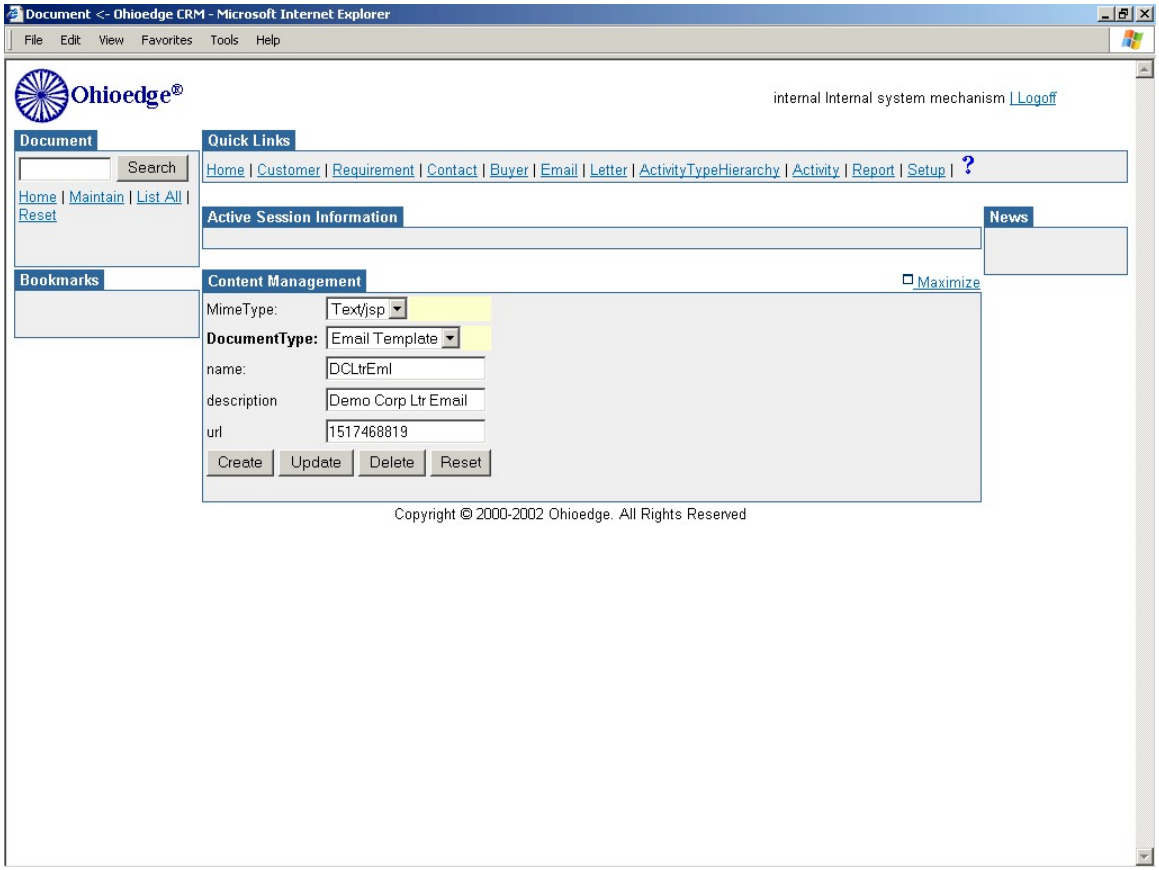


Figure 11.

Ohioedge Publisher Component Setup

30. JSP Scripts

Location: @Setup->JSP Scripts->Maintain or ListAll-><record to maintain>

Dependency: None.

The JSP Scripts business component specifies JSP scripts that expose variable data to documents associated with it.

Example:

Name: Contact

Description: Contact Information Script

JSP Script:

```
<%@ page import="javax.ejb.*, javax.rmi.*, java.rmi.*, javax.naming.*" %>
<jsp:useBean id="sessionBean" scope="session" class="com.ohioedge.j2ee.sales.SessionBean"/>
<%
String organizationName = "";
try {
com.ohioedge.j2ee.api.org.OrganizationBean org = sessionBean.getOrganization();
if (org != null)
organizationName = org.getOrganizationName();
}
catch(Exception e){}
String employeeNamePrefix = "";
String employeeFirstName = "";
String employeeLastName = "";
String employeeNameTitleDescription = "";
try {
com.ohioedge.j2ee.api.org.person.EmployeeBean emp =  sessionBean.getEmployee();
if (emp != null) {
employeeNamePrefix=emp.getNamePrefix().getName();
employeeFirstName=emp.getFirstName();
employeeLastName=emp.getLastName();
employeeNameTitleDescription=emp.getNameTitle().getDescription();
}
}
catch(Exception e){}
String customerName = "";
String customerAddressLine1 = "";
String customerAddressLine2 = "";
String customerCity ="";
com.ohioedge.j2ee.api.address.ejb.State oCustomerState =  null;
String customerState = "";
String customerZip = "";
try {
com.ohioedge.j2ee.api.org.cust.CustomerBean cust = sessionBean.getCustomer();
if (cust != null) {
customerName = cust.getCustomerName();
customerAddressLine1 = cust.getAddressLine1();
customerAddressLine2 = cust.getAddressLine2();
customerCity =cust.getCity();
oCustomerState =  cust.getState();
if (oCustomerState != null)
customerState = oCustomerState.getDescription();
customerZip = cust.getZip();
}
}
catch(Exception e){}
String customerPersonNamePrefixName = "";
String customerPersonFirstName = "";
String customerPersonLastName = "";
try {
com.ohioedge.j2ee.api.org.cust.person.CustomerPersonBean cp = sessionBean.getCustomerPerson();
if (cp != null) {
customerPersonNamePrefixName =
cp.getNamePrefix().getName();
customerPersonFirstName =
cp.getFirstName();
```

```
customerPersonLastName =
cp.getLastName();
}
}
catch(Exception e){}
String contactDate = "MM/dd/yy";
try {
com.ohioedge.j2ee.api.org.proc.ActivityBean con = sessionBean.getActivity();
if (con != null)
contactDate = (new java.text.SimpleDateFormat("MM/dd/yy")).format(con.getActivityDate());
}
catch(Exception e) {
e.printStackTrace();
}
String today = (new java.text.SimpleDateFormat("MMM dd, yyyy")).format(new java.util.Date());
%>
```

31. HTML Letter Templates

Location: @Setup->HTML Templates->Maintain or ListAll-><record to maintain>

Dependency: JSP Scripts.

The HTML Letter Template business component specifies letter templates that publish data in the web-browser.

Example:

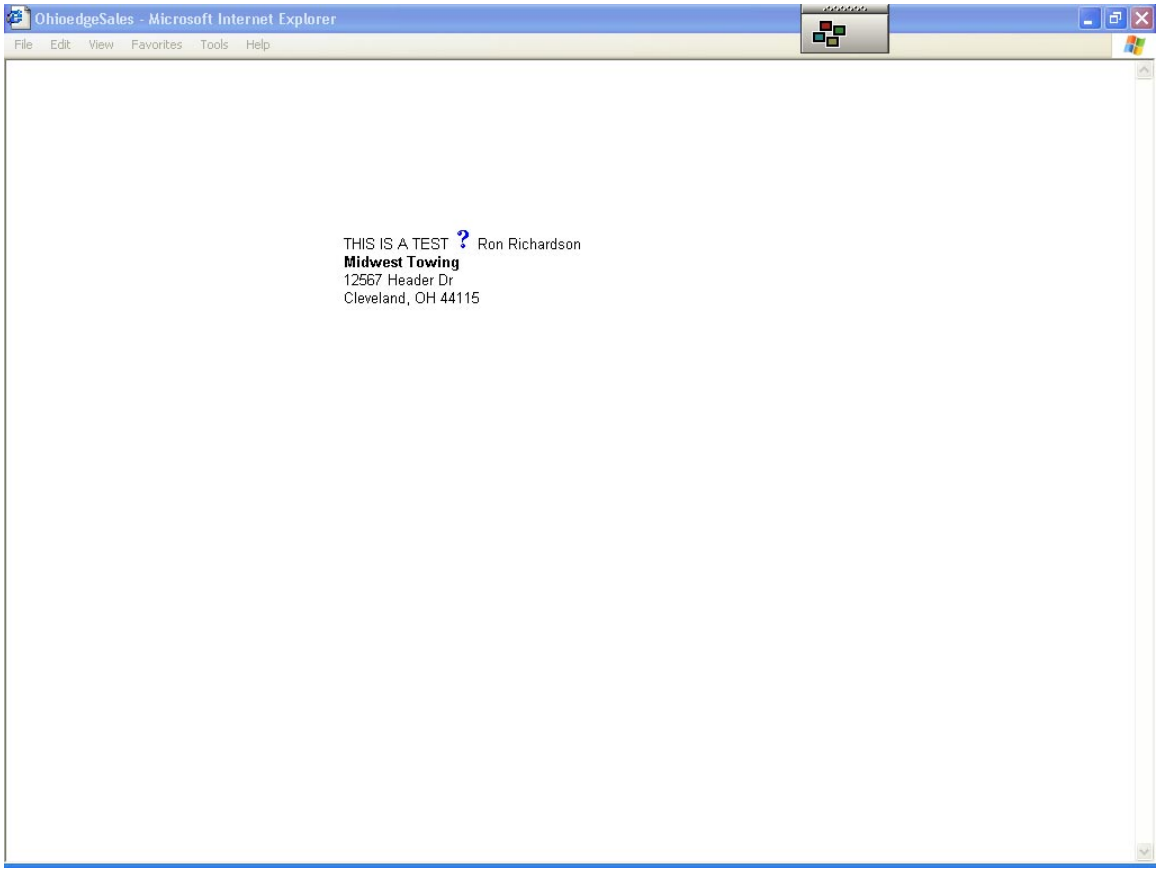
LetterTemplateName: OeInfoEnv

Description: Ohioedge envelope template

JSP Script: Contact Information Script

Template URL: c:\ohioedge\dist\crm\templates\oeenv.html

```
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows-1252">
<TITLE>Envelope # 10</TITLE>
<LINK REL="stylesheet" TYPE="text/css" HREF="css/scm.css"></LINK>
</HEAD>
<BODY>
<SPAN CLASS="envelope"> THIS IS A TEST
<IMG SRC="images/help.gif">
<%=customerPersonFirstName%> <%=customerPersonLastName%><BR>
<STRONG><%=customerName%></STRONG><BR>
<%=customerAddressLine1%><BR>
<%=customerCity%>, <%=customerState%> <%=customerZip%><BR>
</SPAN>
</BODY>
</HTML>
```



At runtime the tags are replaced by the real values. Figure 12 shows the envelope output.

32. Email Templates

Location: @Setup->Email Templates->Maintain or ListAll-><record to maintain>

Dependency: JSP Scripts.

Email Templates are email templates that publish data in the web-browser.

Example:

EmailTemplateName: OeInfoTmpt

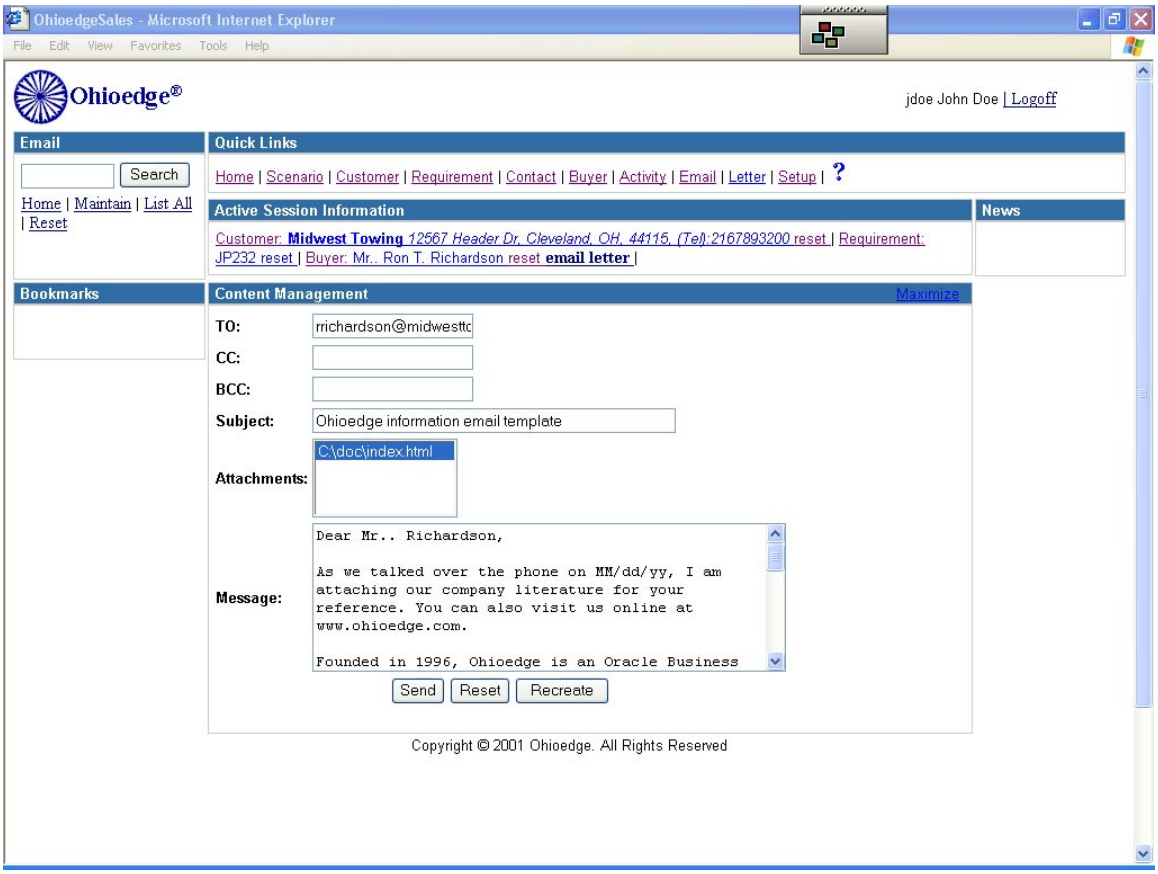
Description: Ohioedge Information Template

Script: Contact Information Script

Template:

```
Dear <%=customerPersonNamePrefixName%>. <%=customerPersonLastName%>,
As we talked over the phone on <%=contactDate%>, I am attaching our company literature for your reference. You can also visit us online at www.ohioedge.com.
Founded in 1996, Ohioedge is an Oracle Business Partner for the last two years. We are proud to have BP Oil, BP Chemicals, Roulston & Co., and R.B. Manufacturing as some of our clients. Our key deliverables include:
1. Custom applications (or components) using Java, XML, and Oracle technology
```

2. OhioedgeSales - an online sales/CRM/workflow application. (To find out more about OhioedgeSales, please take an online tour at http://www.ohioedge.com/scm/tour_1_intro.html)
I hope to have the opportunity to serve you in meeting your e-business, custom programming, and database programming requirements.
Truly yours,
<%=employeeFirstName%> <%=employeeLastName%>
<%=employeeNameTitleDescription%>
Attached: Ohioedge literature; OhioedgeSales literature



Attachments can also be added to the templates using the Email Template Setup. At runtime the tags are replaced by real values. Figure 13 shows how the email looks at runtime.

Ohioedge Organization Component Setup

33. Organization

Location: @Setup->Organization

Dependency: State.

The Organization business component specifies organizations.

34. Employee

Location: @Setup->Employee

Dependency: NamePrefix, NameSuffix, NameTitle.

The Employee business component specifies organizational employees.

35. State

Location: @Setup->State

Dependency: None.

The State business component defines US states.

36. NamePrefix

Location: @Setup->NamePrefix

Dependency: None.

The NamePrefix business component specifies name prefixes, such as, Mr., Ms., etc.

37. NameSuffix

Location: @Setup->NameSuffix

Dependency: None.

The NameSuffix business component specifies name suffixes, such as, Jr., II, etc.

38. NameTitle

Location: @Setup->NameTitle

Dependency: None.

The NameTitle business component specifies name titles, such as, President, IT Manager, etc.

Ohioedge Customer Component Setup

39. BuyerRole

Location: @Setup->BuyerRole

Dependency: None.

The BuyerRole business component specifies the roles of buyers. For example, Decision Maker, Evaluator, etc.

40. RequirementStatus

Location: @Setup->RequirementStatus

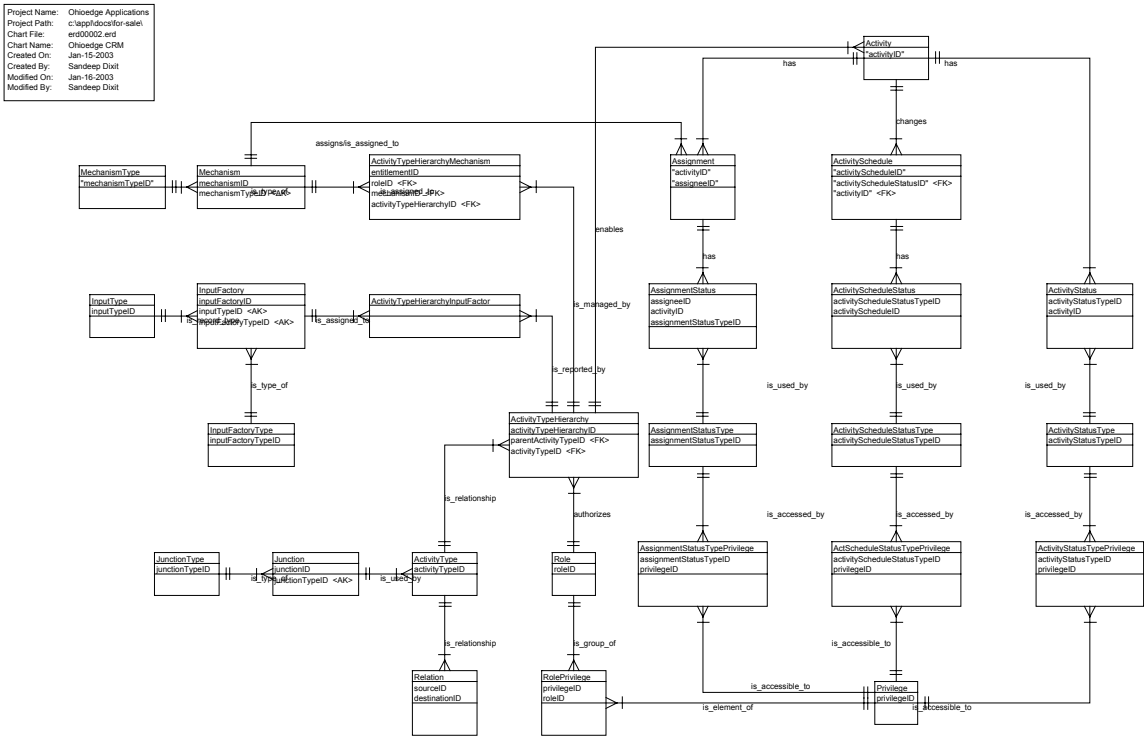
Dependency: None.

The RequirementStatus business component specifies the statuses of a requirement. For example, Open or Closed. Mainly used for the reporting purposes.

Summary

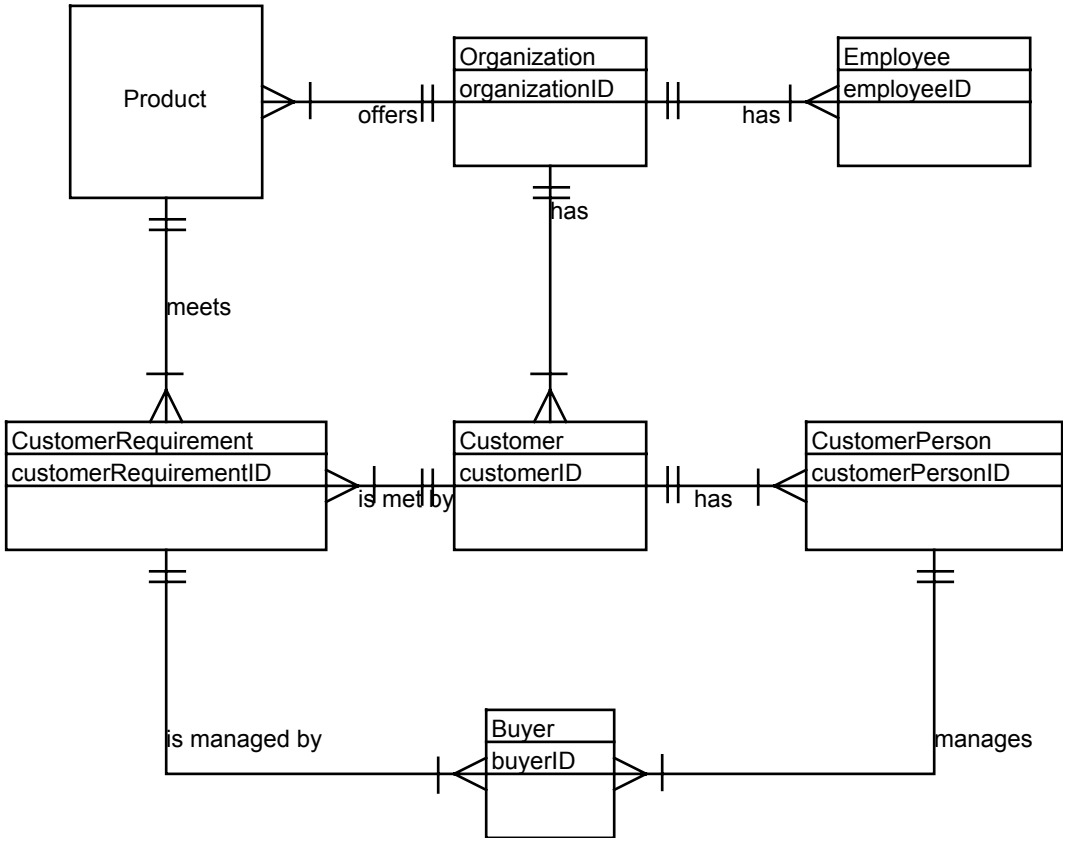
The setup and configuration of Ohioedge CRM consists of setup and configuration of the components it is composed of, namely – Ohioedge Organization, Ohioedge Customer, Ohioedge Publisher, and Ohioedge Workflow. This guide provides a detail explanation of the functionality of the business components requiring setup, their dependencies and relationships, and sample setup data. This guide is intended for the administrators responsible for configuring Ohioedge CRM for their organization-specific business requirements and complements Ohioedge CRM Administrator's Setup Guide.

Appendix A: Ohioedge Workflow Entity Relationship Diagram



Appendix B: Ohioedge J2eeCustomer Entity Relationship Diagram

Project Name:	Ohioedge Applications
Project Path:	c:\appl\docs\for-sale\
Chart File:	erd00006.erd
Chart Name:	Ohioedge J2eeCustomer
Created On:	Jan-16-2003
Created By:	Sandeep Dixit
Modified On:	Jan-16-2003
Modified By:	Sandeep Dixit



Appendix C: Ohioedge J2eePublisher Entity Relationship Diagram