Contract Tree

Solution Story Specification

Author: Navroz Medhora

Version 1.0
Version date: February 28, 2011

TEMPLATE VERSION .3 DATE: January 27, 2003

1. Overview

The Contract Tree (under the ***Contract Maintenance*** tab) displays information of the Contracts laid out in a '*tree*' format for specific levels of the contract. The tree comprises of Company, Contracts and their related Documents, Qualifications, Engines and Data Groups.

Note that the Documents displayed are dependent on the logged in user's security role.

Rates information (which would appear below Data Groups) is not displayed in the contract tree

The Contract Tree can be accessed by clicking on the 'Contract Maintenance' tab or via a Rate Lookup.

When the Contract Tree is accessed through the Contract Maintenance tab, the Contracts displayed are dependent on the selected business group chosen by the user after logging in and their preferred singular Rating Company based on that business group selection. If no Rating Company exists for that particular business group then the tree defaults to displays the company with id 'NO_COMPANY' at the top level in the tree with zero contracts.

In a normal lookup; the Contracts displayed are retrieved from the User's chosen preselected contracts from their last login. If none exist for the particular Rating Company; the User may search for Contracts and add their choice of contracts.

2. System Requirements

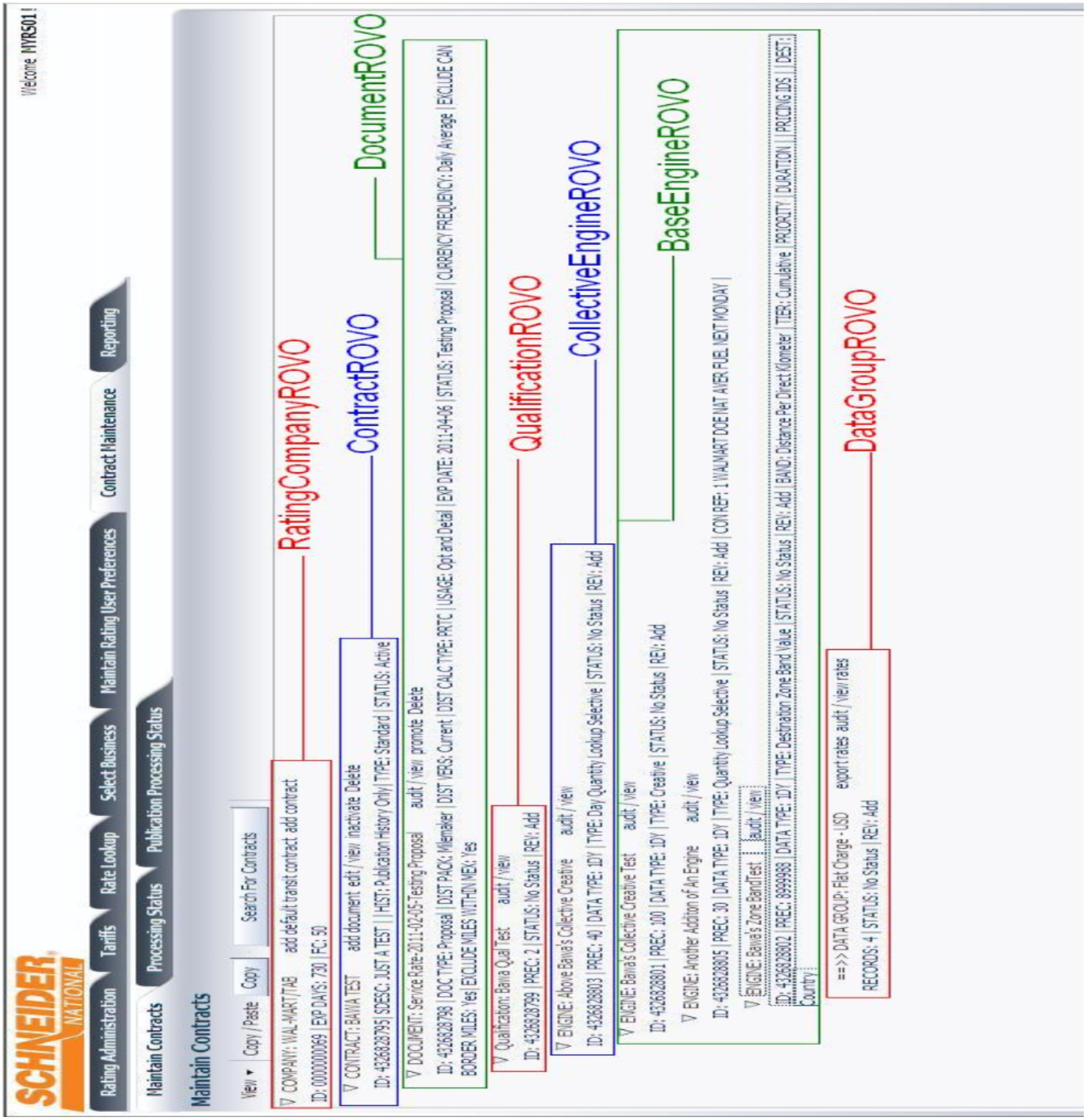
2.1. Completion Tests

For the above defined scope to be considered successful, the following tests need to be executed successfully:

Test Description	Type	Responsible Party

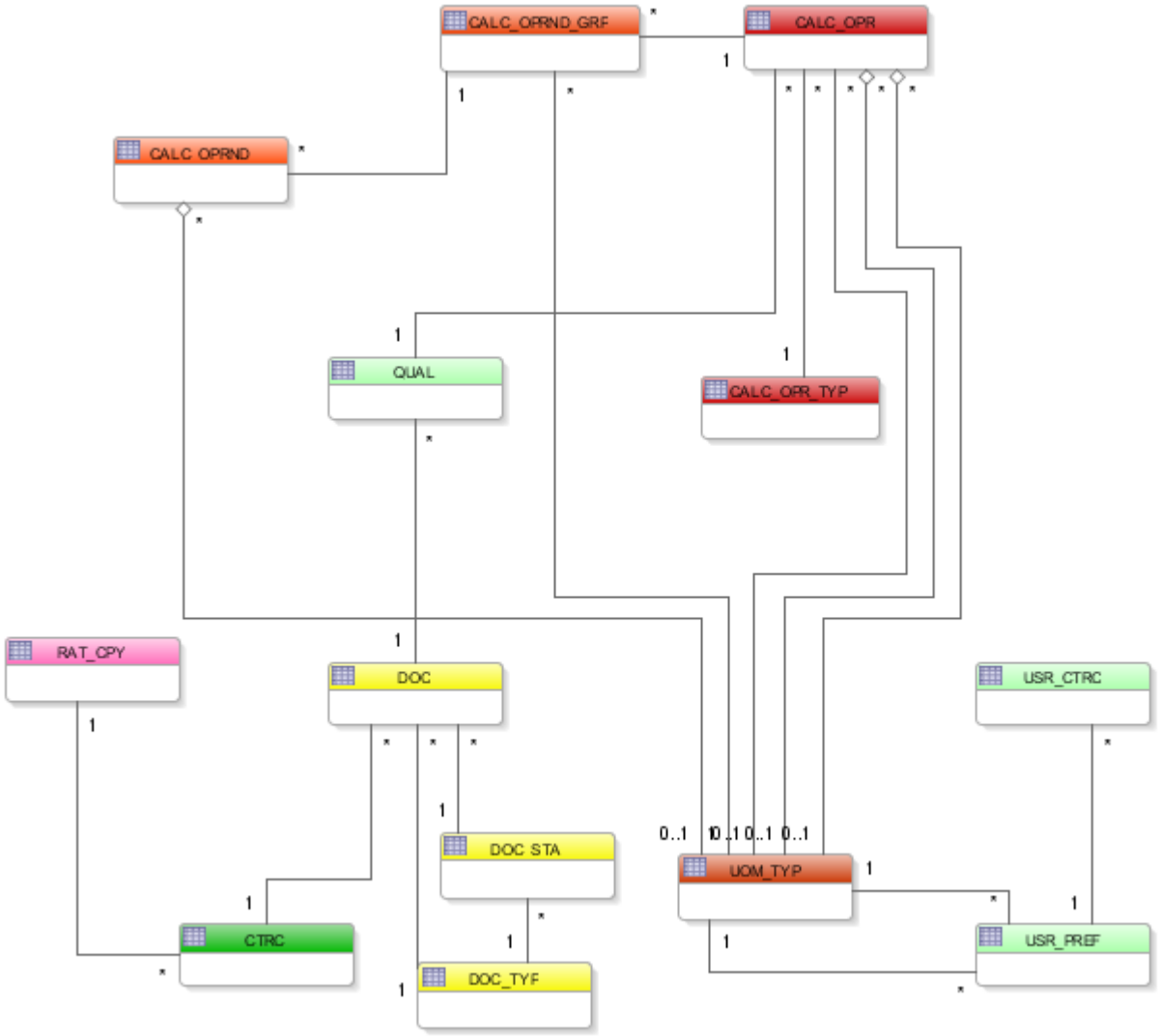
3. Architecture

3.1. Screen Overview



3.2. ERD Diagram

The ERD Diagram below is a snapshot of the tables used for the Contract Tree View Display.



Note that Database Referential Integrity is not directly enforced within all the tables in the schema for performance reasons.

For the Contract Tree tables, each set of tables used are connected to the ones for the level below – as can be noted in the diagram above.

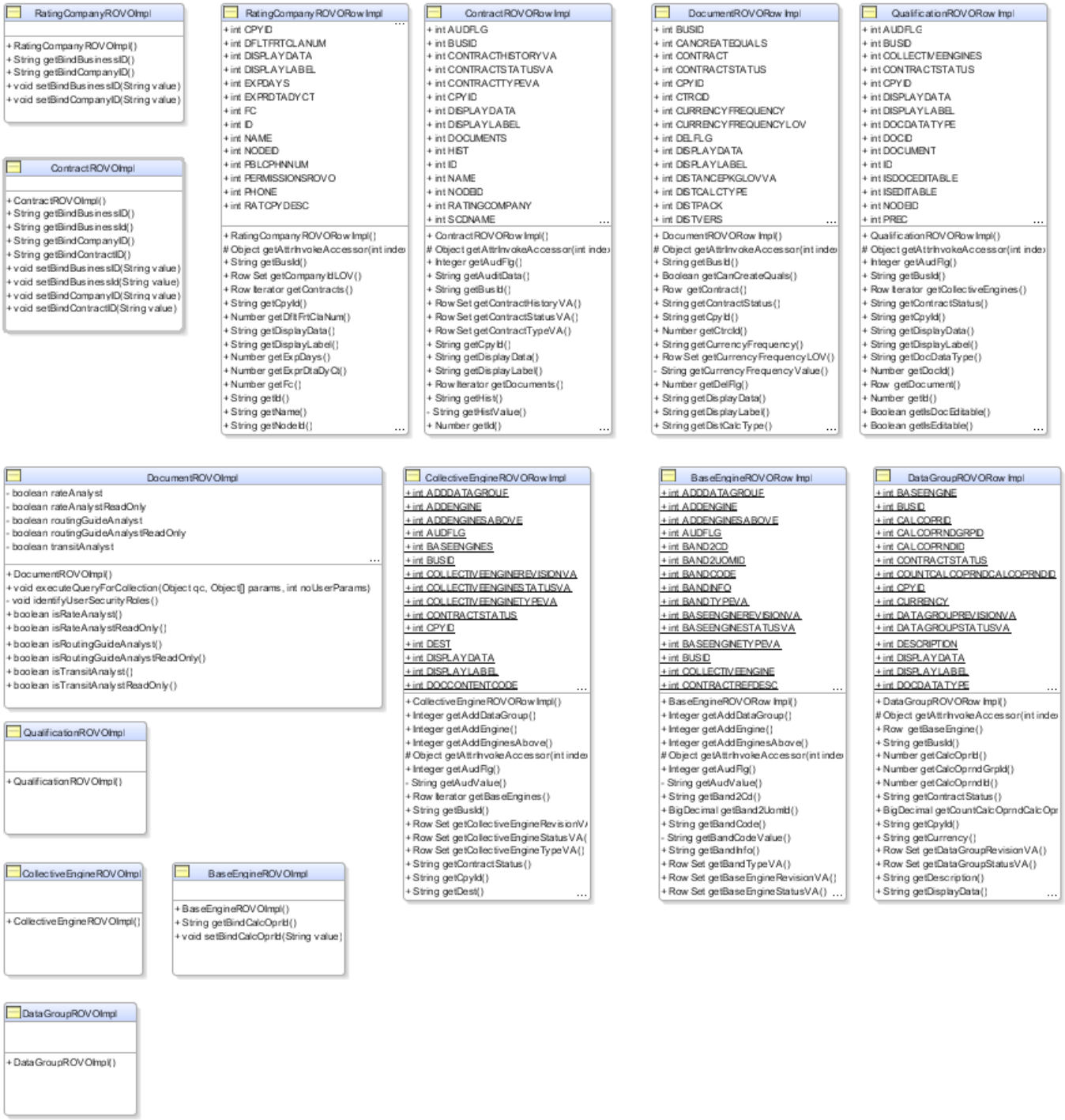
CTRC has a foreign key in BUS_ID and CPY_ID with RAT_CPY table
DOC has a foreign key in CTRC_ID with the CTRC table [DOC table also has foreign keys with DOC_STA and DOC_TYP]
QUAL has a foreign key in DOC_ID with DOC
CALC_OPR [Engine] has a foreign key in QUAL_ID with QUAL table [CALC_OPR table also has a foreign key with CALC_OPR_TYP]

CALC_OPRND_GRP [Data Group] has foreign key in CALC_OPR_ID with CALC_OPR table.
So – even though tables like DOC, QUAL, CALC_OPR, etc may possess attributes like CPY_ID, BUS_ID. These attributes do not form foreign keys to the RAT_CPY table. In effect – the database is denormalized to enable speedy lookups and DML operations; and it is up to the application logic to enforce the referential integrity in the database.

3.3. Class Diagrams

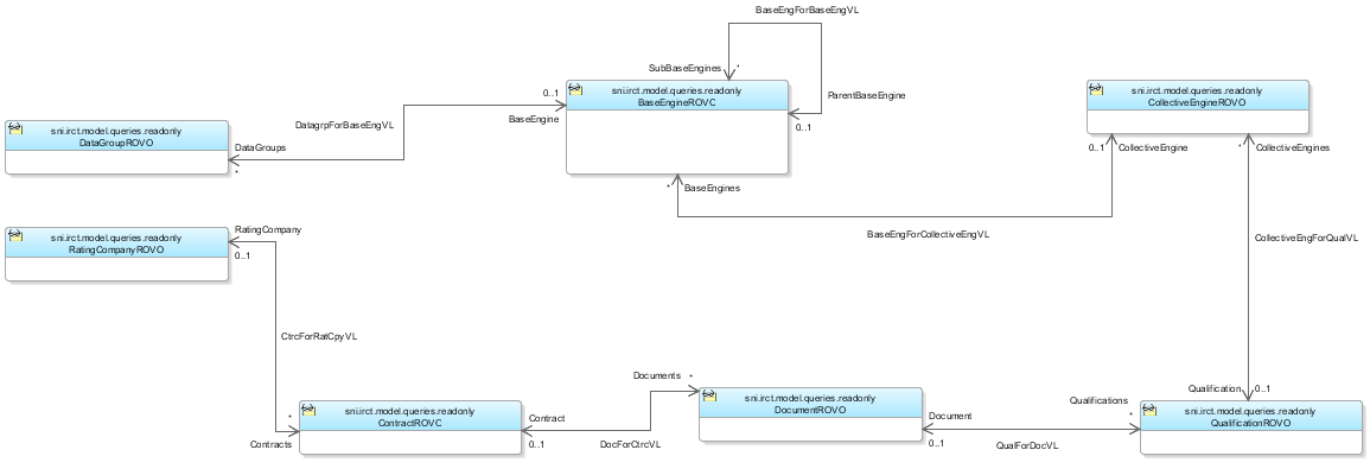
The following class diagram for the sni.irct.view.backing.contracttree.***ContractTreeHandler*** does **NOT** include getter and setter methods for the instance variables **NOR** does it include static fields.

```
ContractTreeHandler
- String companyIdSelectedForCopyPaste
- RichTree contractTree
- RichCommandButton copyButton
- boolean copySelected
- RowKeySet disclosedRowKeys
- boolean displayPasteButtons
- String docDataTypeSelected
- String docDataTypeSelectedForCopy
- String docDataTypeSelectedForPaste
- boolean levelAppropriateForCopy
- boolean levelAppropriateForPaste
- String nodeSelectedForCopyType
- String nodeSelectedForPasteType
- String nodeSelectedId
- String nodeSelectedIdForCopy
- String nodeSelectedIdForPaste
- String nodeSelectedType
- short nodeSelectedTypeVal
- short nodeSelectedTypeValForCopy
- short nodeSelectedTypeValForPaste
- boolean pasteSelected
- RichCommandButton pasteWithButton
- boolean pasteWithDataSelected
- boolean pasteWithTransitSelected
- boolean pasteWithout
- RichCommandButton pasteWithoutButton
- boolean rateAnalyst
- boolean rateAnalystReadOnly
- boolean roleStatusForDocReadOnly
- boolean routingGuideAnalyst
- boolean routingGuideAnalystReadOnly
- Object selectedNode
- boolean transitAnalyst
- boolean transitAnalystReadOnly
+ ContractTreeHandler()
+ Object _invokeMethodExpression(String expr, Class returnType, Class[] argTypes, Object[] args)
+ void addDfltTrnCtrcRow(ActionEvent actionEvent)
+ void cleanUp()
+ void contractTreeMap(ActionEvent evt)
+ void copyNode(ActionEvent actionEvent)
+ void deleteCtrcRow(ActionEvent actionEvent)
+ void expandTree()
- void expandTreeChildrenNode(RichTree rt, FacesCtrlHierNodeBinding node, List<Key> parentRowKey, List<Key> refKey)
+ DCIteratorBinding getDCIterator(String pIterName)
- List<Key> getRowKeyPath(RowKeySet rks, String rowKey)
+ Object getSelectedNode()
+ void getSelection(SelectionEvent evt)
+ void handleRowDisclosure(RowDisclosureEvent rowDisclosureEvent)
- void identifyUserSecurityRoles()
+ void inactivateCtrcRow(ActionEvent actionEvent)
+ void initialize()
+ void pasteSelectedNode(ActionEvent actionEvent)
+ void viewDfltTrnCtrRow(ActionEvent actionEvent)
```

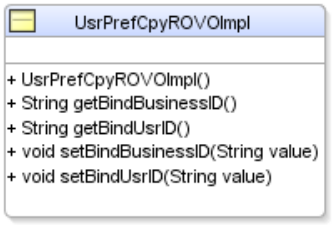



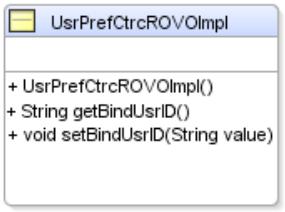
The above image displays the Class Diagram for the ROVOImpl and ROVORowImpl classes *used directly* by the ContractTree display.

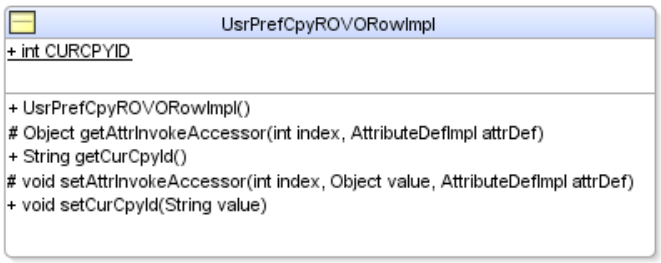
The following diagram displays the relations (including View Links) between the ADF ROV objects

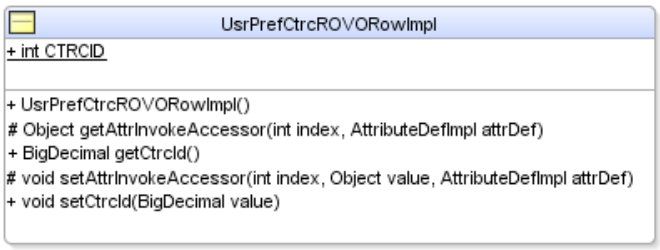


The UsrPrefCpyROVO and UsrPrefCtrcROVO help determine which company and which contracts to display – when the user enters the contract tree in the normal flow --- i.e. **not via a Rate Lookup.**





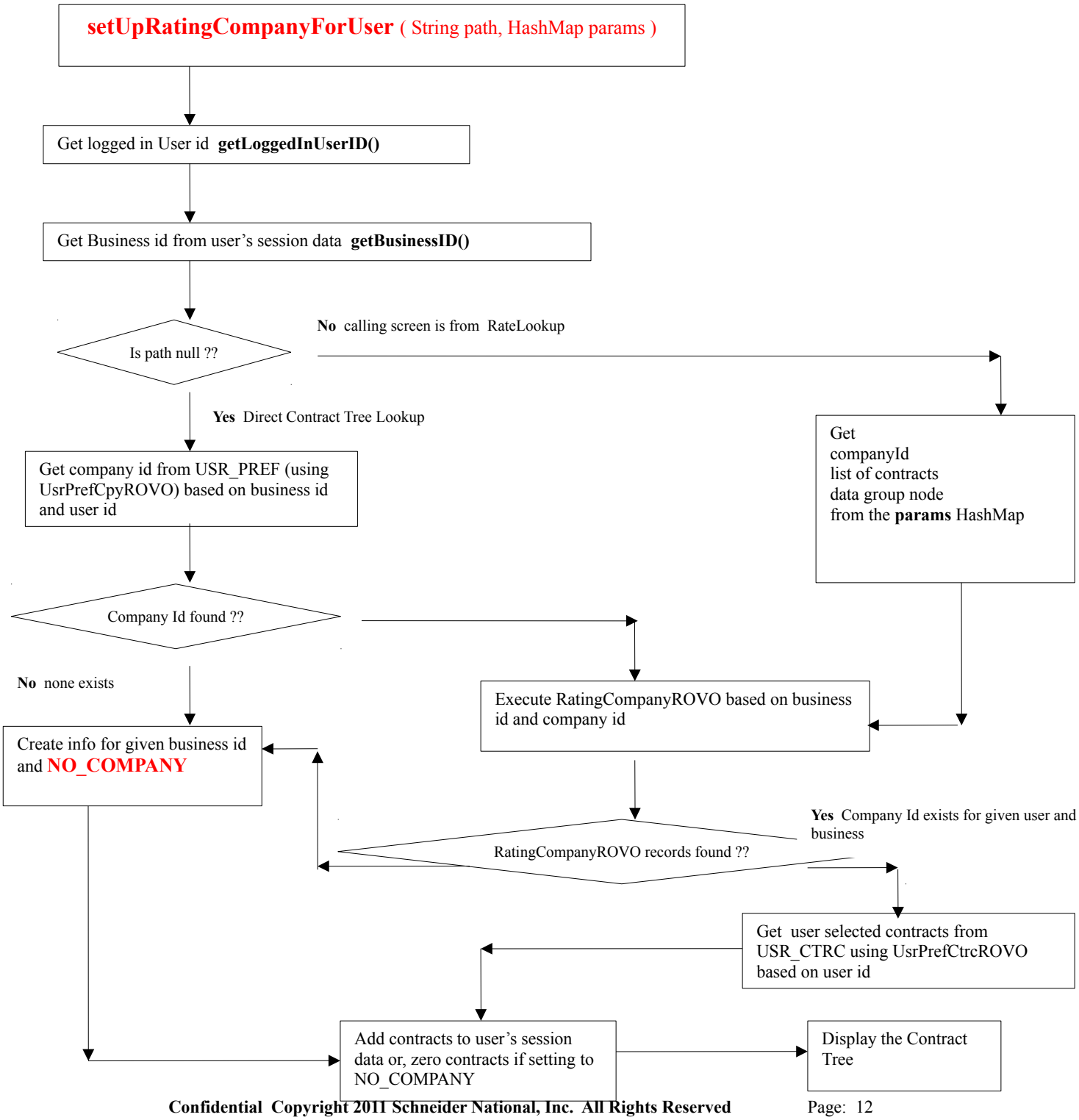




The UserSecurityInfo interface and UserSecurityInfoImpl class – a Singleton class – help in determining the security roles of the logged in user:

3.4. Sequence Diagram / Flow Charts

The ***setUpRatingCompanyForUser*** is the entry point and the default activity into the contract maintenance task flow



The operations for creating a NO_COMPANY record for the user in the USR_PERF table when matches are not found in the database for the given user and business group involves the ***cleanUpUsrPrefData*** method.

The purpose of this utility method is to perform the following tasks:

1. Updates the USR_PREF record for the given user setting CUR_CPY_ID to NO_COMPANY and CUR_BUS_ID to busId
2. Deletes the existing user selected contracts if any from the USR_CTRC table
3. Executes the RatingCompanyROVO with the changes so the tree is refreshed
4. Sets the user session data for CONTRACT_IDS. We set the value to a SPACE.

The following steps are conducted by the method:

cleanUpUsrPrefData(String usrId, String busId)

1 → updateUsrPrefInfo(String pusrId, String pcpyId, String pbusId)

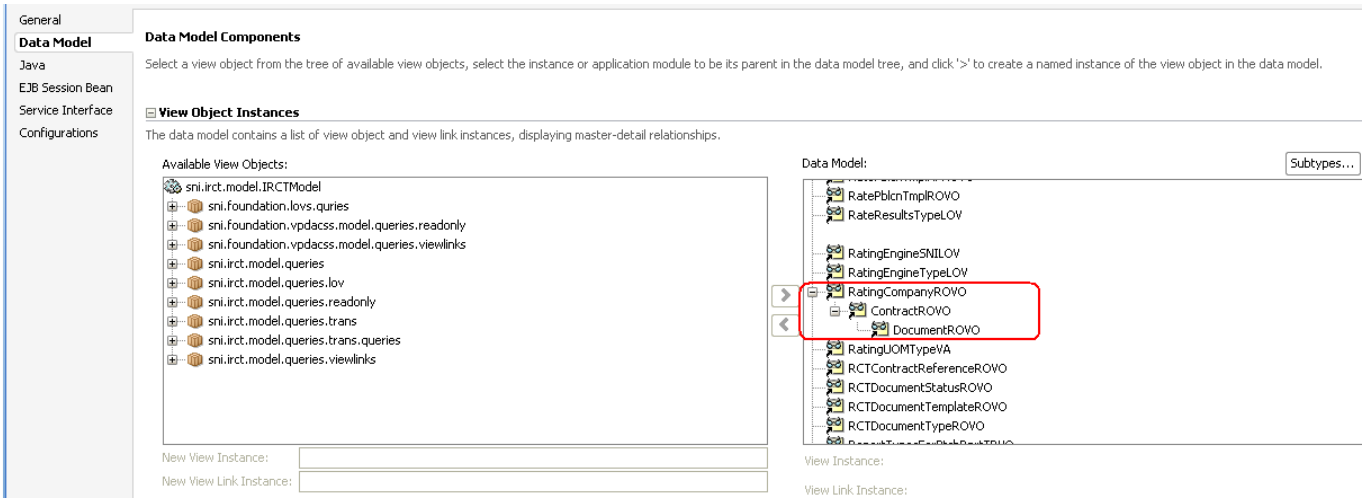
2 → deleteExistingUsrContracts(String pusrId)

3 → Executes the RatingCompanyROVO with the changes so the tree is refreshed

4 → Sets the user session data for CONTRACT_IDS to " "

Since the setUpRatingCompanyForUser in the IRCTRateCompanyService calls upon the RatingCompanyROVO, ContractROVO (and consequently its associated View Links); only the object instances upto DocumentROVO need appear within its Data Model.

Data Model for IRCTRateCompanyService



To continue the process of modularization and further stream line the IRCTRateCompanyService Application Module: it is expected that in future; the ROVO objects used by the Contract Maintenance (and the related screens for the Contract Tree) will be moved to a new project – the *IRCTContractMaintenance* project

3.5. Security Considerations

The Contract Maintenance tab and related sub screens including the Contract Tree can only be accessed by the following Security Roles:

- ***RSRateAnalyst***
- ***RSRateAnalystReadOnly***
- ***RSRoutingGuideAnalyst***
- ***RSRoutingGuideAnalystReadOnly***
- ***RSTransitAnalyst***
- ***RSTranistAnalystReadOnly***
- ***RSPublicationAnalyst***
- ***RSPublicationInitiator***

These two roles cannot access the Contract Tree View but have access to the *Publication* tabs

These access rights have been granted from the ***jazn-data.xml*** file for the
contractmaintenance-config task flow
contractmaintenance.jsff screen
contractmaintenance_config_contractmaintenance_config_setUpRatingCompanyForUserPageDef
contracttree.jsff screen
maintaincontractshome task flow

Based on the security role of the user, only certain actions can be performed and certain document types are visible.

The following table delineates the restrictions and access grants applicable for the security roles in connection with the Contract Tree.

Role	Task	OID Role	Visible Document Data Types
Rate Analyst	Maintain Contracts (Rate and Lookup Documents)	RSRateAnalyst	LKUP, PRD, SRV
Rate Analyst Read Only	Maintain Contracts (Rate and Lookup Documents) No Create/Update capabilities	RSRateAnalystReadOnly	LKUP, PRD, SRV
Transit Analyst	Maintain Contracts (Transit and Lookup Documents)	RSTransitAnalyst	LKUP, TNST
Transit Analyst View Only	Maintain Contracts (Transit and Lookup Documents) No Create/Update capabilities	RSTransitAnalystReadOnly	LKUP, TNST
Routing Guide Analyst	Maintain Contracts (Routing Guide and Lookup Documents)	RSRoutingGuideAnalyst	LKUP, RTNG
Routing Guide Analyst View Only	Maintain Contracts (Routing Guide and Lookup Documents) No Create/Update capabilities	RSRoutingGuideAnalystReadOnly	LKUP, RTNG

The following table pivots the above information and describes which document data types can be accessed by which user security roles

Doc Data Types	***Full Name***	***Can be seen by***
LKUP	Lookup	RSRateAnalyst RSRateAnalystReadOnly RSTransitAnalyst RSTransitAnalystReadOnly RSRoutingGuideAnalyst RSRoutingGuideAnalystReadOnly
PRD	Product Rate	RSRateAnalyst RSRateAnalystReadOnly
RTNG	Routing Guide	RSRoutingGuideAnalyst RSRoutingGuideAnalystReadOnly
SRV	Service Rate	RSRateAnalyst RSRateAnalystReadOnly
TNST	Transit	RSTransitAnalyst RSTransitAnalystReadOnly

Depending on whether the user's security role is defined as ***Analyst or ***Analyst*ReadOnly* the links displayed on the contract tree may vary.

So for example – a user with RoutingGuideAnalyst security role may have "add ***", "edit / view" and "delete" links visible for the contract tree for LKUP and RTNG document data types (and for their sub levels) but would have no access to SRV, PRD and TNST document data types. The links displayed are dependent on the values of corresponding flags in the ***DOC_STA*** table for the applicable document.
A RoutingGuideAnalystReadOnly security role will ***not*** have "add ***", "edit / view" and "delete" links visible for the contract tree for LKUP and RTNG document data types (and for their sub levels); instead they would have "audit/view" and "view" links available to them. They also would have no access to SRV, PRD and TNST document data types.

Moreover, overriding the add, edit and delete capabilities defined by the flags in the DOC_STA table is the contract status – if the contract status is 'ACTV' for the particular contract as we go down the tree - then the "add", "edit/view" and "delete" links may be displayed. However if the contract status (taken from the ***STA_CD***) value in the ***CTRC*** table is '***INAC***', then only the 'view' links are displayed.

The ContractTreeHandler during instantiation calls the ***identifyUserSecurityRoles*** method within its constructor to identify the roles applicable for the user. This method uses the sni.irct.model.util. ***UserSecurityRoleInfo*** interface to determine the above mentioned roles.

Within the ContractTreeHandler class the following instance fields of ***boolean*** type (and their associated implications for assigned User Security Roles) are used:

Boolean instance within ContractTreeHandler	User Security Role
rateAnalyst	RSRateAnalyst
rateAnalystReadOnly	RSRateAnalystReadOnly
transitAnalyst	RSTransitAnalyst
transitAnalystReadOnly	RSTransitAnalystReadOnly
routingGuideAnalyst	RSRoutingGuideAnalyst
routingGuideAnalystReadOnly	RSRoutingGuideAnalystReadOnly

It is also expected that ***Analyst and ***AnalystReadOnly roles are mutually exclusive for a User.
In other words a user ***cannot*** possess both the RSTransitAnalyst **and** RSTransitAnalystReadOnly roles (for example).

Dynamic ADF command links in the contract tree include these boolean values within conditions for their ***render*** attribute.

It is expected for the ***Contract Maintenance*** tab to be displayed the user belongs to at least one of the above six defined security roles. Also having logged in access exists for the user id. Moreover, a selection of the business group has already been undertaken by the user (in the event the user belongs to more than one business group)

So it is expected that the methods ***getLoggedInUserID*** and ***getBusinessID*** in the IRCTRateCompanyServiceImpl application module will not return nulls or an empty String.

The entry point into the Contract Maintenance task flow is the ***setUpRatingCompanyForUser*** method. Its definition for this method is also protected by the Security Roles as described above.

The method signature is as follows:

```
public void setUpRatingCompanyForUser(String path, HashMap params)
```

3.6. Contract Tree displayed via a Rate Lookup

When the parameters – of the String *path* and the HashMap *params* – are both **NOT** null; then we are entering the Contract Tree via a Rate Lookup. In this case; the value for the String path is "***lookup***" and the HashMap params comprises the values for the ***company, contract and the data group id*** for which level the contract tree will be expanded to.

String path [if not null]

IRCTRateCompanyConstants.TASK_FLOW_PARAM_LKPRTE_CONST	lookuprate

HashMap params [if not null has the following keys]

Key code		Value type	Info
IRCTRateCompanyConstants.TASK_FLOW_PARAM_COMPANY_ID	companyID	String	ID of Company to display
IRCTRateCompanyConstants.TASK_FLOW_PARAM_CONTRACT_ID	contractID	List<String>	List of Contracts
IRCTRateCompanyConstants.TASK_FLOW_PARAM_DATA_GROUP_ID	dataGrpID	String	Node Id to expand to

3.7. Regular Contract Tree Display

Here both the parameters to the `setUpRatingCompanyForUser` method are **null**. The selection for the top level Rating Company is dependent on the information taken from **USR_PREF** table via the ***UsrPrefCpyROVO*** view which provides us with the company id, based on the user's chosen business group. This company id is then used as a bind variable in the RatingCompanyROVO view which retrieves information from the **RAT_CPY** table.

See the above diagram showing the process flow from `setUpRatingCompanyForUser`

3.8. Contract Tree ReadOnly Views

Depending on the hierarchical level in the tree; *a separate read only view* is used to gather the information for display.

All the read only views used to display the contents of the Contract Tree are located in the ***IRCTModel*** project in the ***sni.irct.model.queries.readonly*** package.

Each ReadOnly ViewObject in the Contract Tree **ALWAYS** has the following **transient** **attributes**:

NodeId	A String to help determine the node in the tree – the value includes the name of the level plus the id in the view
DisplayLabel	A String which occurs beside each node
DisplayData	A String which occurs below each node

The values for the above attributes are determined in the getter methods for their respective ***ROWImpl*** implementations

getNodeId() → for each level in the tree will return a String. The contents of the String are the name of the level concatenated with an underscore "_", concatenated with the value of the ID attribute for the row.

Level	***getNodeId()value***	***Implementation Class***
Company	company_IDoftheCompany	RatingCompanyROVORowImpl
Contract	contract_IDoftheDocument	ContractROVORowImpl
Document	document_IDoftheDocument	DocumentROVORowImpl
Qualification	qualification_IDoftheQualification	QualificationROVORowImpl
Collective Engine	collectiveengine_IDoftheEngine	CollecitiveEngineROVORowImpl
Base Engines	baseengine_IDoftheEngine	BaseEngineROVORowImpl
Data Group	datagroup_IDoftheDataGroup	DataGroupROVORowImpl

3.8.1. Company Level → RatingCompanyROVO, UsrPrefCpyROVO & UsrPrefCtrcROVO

The primary view used at the Company level for displaying the contents is the ***RatingCompanyROVO***. which extracts its information from the snirate.**RAT_CPY** table.

View Accessors used in the RatingCompanyROVO are

- CompanyIdLOV
- PermissionsROVO

From the ***setUpRatingCompanyForUser*** method: the business id is retrieved via the ***getBusinessID()*** method and the user id is retrieved via the ***getLoggedInUserId()*** methods.

When the Contract Tree is accessed in the regular way, via the 'Maintain Contracts' tab; first the ***UsrPrefCpyROVO*** is executed to identify which company [***CUR_CPY_ID*** *from the* ***USR_PREF*** table for the logged in user] is to be used for display at the top of the tree given the user's selected business id.

Next, from the ***UsrPrefCtrcROVO***, the user's preferred contracts are chosen from the list of CTRC_ID in the USR_CTRC table for the logged in user. Once these contracts are identified; they are placed into the user's session data under the key: IRCTRateCompanyConstants.**CONTRACT_IDS**. *Note that if no user selected contracts exist – then we insert a blank space → " " → as the value.*

The SQL query used in **UsrPrefCpyROVO** is:

```
select cur_cpy_id
from snirate.usr_pref
where cur_bus_id = :BindBusinessID
and usr_id = :BindUsrID
```

Bind Variable for UsrPrefCpyROVO	Value
BindBusinessID	adf.userSession.userData.businessID
BindUsrID	adf.context.securityContext.userName

The SQL query used in **RatingCompanyROVO** is:

```
select
    c.CPY_ID ID,
    c.EXPR_DTA_DY_CT EXP_DAYS,
    c.DFLT_FRT_CLA_NUM FC,
    c.PBLC_PHN_NUM PHONE,
    c.BUS_ID,
    c.RAT_CPY_DESC NAME
from snirate.RAT_CPY c
where c.BUS_ID = :BindBusinessID
and c.CPY_ID = :BindCompanyID
```

Bind Variable for RatingCompanyROVO	Value
BindBusinessID	adf.userSession.userData.businessID
BindCompanyID	1. Normal Contract Tree Lookup ← value of CUR_CPY_ID from the UsrPrefCpyROVO query above 2. Contract Tree displayed after Rate Lookup → value associated with IRCTRateCompanyConstants.**TASK_FLOW_PARAM_COMPANY_ID** key in the ***params*** HashMap parameter in the ***setUpRatingCompanyForUser method***

The SQL query used in **UsrPrefCtrcROVO** is:

```
select CTRC_ID
from snirate.USR_CTRC
where USR_PREF_ID in (
  select USR_PREF_ID
  from snirate.USR_PREF
  where usr_id = :BindUsrID
)
```

Bind Variable for UsrPrefCtrcROVO	Value
BindUsrID	adf.context.securityContext.userName

3.8.2. ContractROVO

The tables used in the ContractROVO is the SNIRATE.***CTRC*** table.

The view has two bind variables; the business id and the list of contracts taken from the session data (which is set by the setUpRatingCompanyForUser method).

The business id is dependent on the user selected business group and is stored in the ***adf.userSession.userData.businessID*** session variable.

The list of contract ids comes from the ***adf.userSession.userData.contractIds*** session variable. It is a *comma-delimted* string.

The SQLquery used in **ContractROVO** is:

```
select CPY_ID,
       BUS_ID,
       CTRC_ID ID,
       HST_MTHD_CD HIST,
       CTRC_CD TYPE,
       STA_CD STATUS,
       CTRC_DESC NAME,
       CTRC_SCD_DESC SCD_NAME,
       AUD_FLG
from SNIRATE.CTRC
where BUS_ID = :BindBusinessId
and (:BindContractID IS NULL OR
      (:BindContractID is NOT NULL
       AND CTRC_ID IN (
           SELECT *
           FROM
TABLE(CAST(sni.sni_gn_in_string_list(:BindContractID) AS
           sni.table_of_varchar)))
      )
    )
order by NAME
```

Bin	e
BindBusinessI	essID
BindContractID	actIds

The ContractROVO view has three View Accessors used as LOVs for three of its attributes:

View Accessor Name	***Static LOV / SNI Foundation LOV***	***Attribute Applicable***	***Application Code [SNI]***	***Domain Name [SNI]***	***Method in ContractROVORowImpl to retrieve 'decoded' value***
ContractHistoryVA	SNI Foundation LOV	Hist	RT	RATING_CONTRACT_HIST	getHistValue()
ContractStatusVA	SNI Foundation LOV	Status	RT	RATING_CONTRACT_STATUS	getStatusValue()
ContractTypeVA	SNI Foundation LOV	Type	RT	RATING_CONTRACT_TYP	getTypeValue()

3.8.3. DocumentROVO

The DocumentROVO view gets its information from the following tables:

- SNIRATE.**DOC**
- SNIRATE.**DOC_STA**
- SNIRATE.**DOC_TYP**
- SNIRATE.**V_DIST_GEO_PKG**

The documents are displayed in descending order of the documents content code and effective date [DOC_CTNT_CD and EFF_DT attributes in DOC table].

The SQL query used in DocumentROVO is:

```
select
        doc.CTRC_ID,
        doc.CPY_ID,
        doc.BUS_ID,
        doc_sta.PROM_FLG,
        doc_typ.DOC_CTNT_CD DOC_CONTENT_CODE,
        doc.EFF_DT,
        doc.DOC_ID ID,
        doc.DOC_TYP_ID,
        doc_typ.DOC_CD DOC_TYPE,
        geo.GEO_PKG_CD DIST_PACK,
        geo.PKG_VERS_CD DIST_VERS,
        geo.DIST_CALC_MTHD_CD DIST_CALC_TYPE,
        doc.DOC_CTNT_USG_CD USAGE,
        doc.EXPR_DT EXP_DATE,
        doc_sta.DOC_STA_CD STATUS,
        doc.DOC_STA_ID,
        doc.CRCY_CNV_RULE_CD CURRENCY_FREQUENCY,
        doc.EXCL_CNDNXBRDR_FLG EXCL_CAN_BORDER_MILES,
        doc.EXCL_MEX_MI_FLG EXCL_MILES_WITHIN_MEX,
        doc_sta.REQ_CRE_FLG,
        doc_sta.UPD_FLG,
        doc_sta.DEL_FLG,
        doc_sta.UPD_EFF_DT_FLG,
        doc_sta.UPD_EXPR_DT_FLG
from
        SNIRATE.DOC doc,
        SNIRATE.DOC_STA doc_sta,
        SNIRATE.DOC_TYP doc_typ,
        SNIRATE.V_DIST_GEO_PKG geo
where
       doc.DOC_TYP_ID = doc_typ.DOC_TYP_ID
and doc.DIST_PKG_ID = geo.GEO_PKG_ID
and doc.DOC_STA_ID = doc_sta.DOC_STA_ID
order by doc_typ.DOC_CTNT_CD, doc.EFF_DT
```

The DocumentROVO has no explicit bind variables. The Documents for the Contract are ordered by DOC_CTNT_CD and EFF_DT.

The DocumentROVO view has the following View Accessors used as LOVs for three of its attributes:

View Accessor Name	***Static LOV / SNI Foundation LOV***	***Attribute Applicable***	***App Code [SNI]***	***Domain Name [SNI]***	***Method in DocumentROVORowImpl to retrieve 'decoded' value***
CurrencyFrequencyLOV	Static LOV	CurrencyFrequency			getCurrencyFrequencyValue()
DistancePackageCalcTypeVA	SNI Foundation LOV	DistCalcType	RT	RATING_DISTANCE_CALC_TYPE	get DistCalcType Value()
DistancePackageVA	SNI Foundation LOV	DistPack	RT	RATING_DISTANCE_PKG	getDistPackValue()
DocumentDataTypeVA	SNI Foundation LOV	DocContentCode	RT	RATING_DOCUMENT_DATA_TYPE	getDocContentCodeValue()
DocumentStatusVA	SNI Foundation LOV	Status	RT	RATING_DOCUMENT_STATUS	getStatusValue()
DocumentTypeVA	SNI Foundation LOV	DocType	RT	RATING_DOCUMENT_TYPE	getDocTypeValue()
UsageLOV	Static LOV	Usage			getUsageValue()

For determining the display of the dynamic command links from within and below the Document level; the DocumentROVO has several ***transient attributes***, apart from the usual ones of Id, DisplayData, DisplayLabel and NodeId***:***

These often connect to Flags within the DOC_STA and DOC_TYP tables. In addition the ContractStatus is a transient variable for these ROVOs.

Additional Transient Variables in DocumentROVO [Flags coming in from DOC_STA table]

Flag used from DOC_STA table	Transient Variable [DocumentROVO]	Purpose
No Flag used – Value is determined by identifying the Contract row by traversing up the tree view using the **getContract** method in the *DocumentROVORowImpl* class	ContractStatus	Its value is either 'ACTV' or "INAC' – depending on which the dynamic links are displayed in the levels below.
REQ_CRE_FLG	CanCreateQuals	To determine whether the '***add qualification***' link may be displayed for ***Analyst role and *active* Contracts
DEL_FLG	IsDeletable	To determine whether the '***delete'*** link may be displayed for ***Analyst role and *active* Contracts
UPD_FLG, UPD_EFF_DT_FLG and UPD_EXPR_DT_FLG	IsEditable	When ***any*** of these flags are **1** – the Document is considered *editable* – and the '***edit / view***' link is displayed for ***Analyst roles or '***audit / view***' link is displayed for ***AnalystReadOnly roles These apply for *active* Contracts
UPD_FLG	IsUpdatable	To determine whether '***edit / view***' should be displayed for ***Analyst roles or whether '***audit / view***' link is displayed for

		***AnalystReadOnly roles for the levels below the Documents node These apply for *active* Contracts
PROM_FLG	IsPromotable	To determine whether the ***'promote'*** link is to be displayed for *active* Contracts

The Documents displayed in the Contract Tree are dependent on the User Security Role as described above in the section – Security Considerations.

The process of ensuring this is done within the DocumentROVOImpl class is in the ***executeQueryForCollection*** method which is overridden to filter out the applicable documents based on the security role. The logged in user's security roles is determined by the method ***identifyUserSecurityRoles*** from within that method – and an additional where-clause is dynamically created on-the-fly based on the user's security role.

```
@Override
public void executeQueryForCollection(Object qc, Object[] params, int noUserParams)   {
            ....
  identifyUserSecurityRoles();

 if (rateAnalyst || rateAnalystReadOnly || transitAnalyst ||
    transitAnalystReadOnly || routingGuideAnalyst ||
    routingGuideAnalystReadOnly)     {

  StringBuffer sb = new StringBuffer("");
  final String DOC_CONTENT_CODE_IN = " DOC_CONTENT_CODE IN ( ";
          ...

  sb.append(DOC_CONTENT_CODE_IN);

  if (rateAnalyst || rateAnalystReadOnly)      {
    sb.append(QUOTE + IRCTRateCompanyConstants.DOCUMENT_TYPE_LKUP + QUOTE_COMMA_QUOTE +
            IRCTRateCompanyConstants.DOCUMENT_TYPE_PRD + QUOTE_COMMA_QUOTE +
            IRCTRateCompanyConstants.DOCUMENT_TYPE_SRV + QUOTE_SPACE);
  }

 if (transitAnalyst || transitAnalystReadOnly) {
   if (rateAnalyst || rateAnalystReadOnly) {
     sb.append(COMMA);
   }

   sb.append(QUOTE + IRCTRateCompanyConstants.DOCUMENT_TYPE_LKUP + QUOTE_COMMA_QUOTE +
            IRCTRateCompanyConstants.DOCUMENT_TYPE_TNST + QUOTE_SPACE);
  }

  if (routingGuideAnalyst || routingGuideAnalystReadOnly) {
    if (rateAnalyst || rateAnalystReadOnly || transitAnalyst || transitAnalystReadOnly) {
      sb.append(COMMA);
    }

    sb.append(QUOTE + IRCTRateCompanyConstants.DOCUMENT_TYPE_LKUP + QUOTE_COMMA_QUOTE +
            IRCTRateCompanyConstants.DOCUMENT_TYPE_RTNG + QUOTE_SPACE);
  }

  sb.append(")");

  setWhereClause(sb.toString());
 }

 super.executeQueryForCollection(qc, params, noUserParams);

 ....

}
```


3.8.4. QualificationROVO

The QualificationROVO view gets its information from the **QUAL** table

The SQL query used in QualificationROVO is:

```
select
     CPY_ID,
     DOC_ID,
     BUS_ID,
     QUAL_ID ID,
     QUAL_DESC,
     PRCDN_NUM PREC,
     STA_CD STATUS,
     RVS_CD REV,
     AUD_FLG
from SNIRATE.QUAL
```

The QualificationROVO view has no explicit bind variables.

The QualificationROVO view has the following View Accessors for two of its attributes

View Accessor Name	*Static LOV / SNI Foundation LOV*	*Attribute Applicable*	*App Code [SNI]*	*Domain Name [SNI]*	*Method in QualificationROVORowImpl to retrieve 'decoded' value*
QualificationRevisionVA	SNI Foundation LOV	Rev	RT	RATING_REVISION_CD	getRevValue()
QualificationStatusVA	SNI Foundation LOV	Status	RT	RATING_STATUS_CD	getStatusValue()

For determining the display of the dynamic command links; the QualificationROVO has the following transient attributes:

Flag used from QUAL or DOC_STA tables	Transient Variable [QualificationROVO]	Purpose
No Flag used – Value is determined by identifying the Contract row by traversing up the tree views using the **getContractStatus** method in the *QualificationROVORowImpl* class	ContractStatus	Its value, either 'ACTV' or 'INAC' is determined by traversing up the rows.
No Flag used – Value is determined by identifying the Document row by traversing up the tree view using the ***getDocDataType*** method in the QualificationROVORowImpl class	DocDataType	To identify the document data type that the qualification belongs to
UPD_FLG from DOC_STA table – Value is determined by identifying the Document row by traversing up the tree view using the ***getIsDocEditiable***	IsDocEditable	To determine whether '***edit / view***' should be displayed for ***Analyst roles or whether '***audit / view***' link is displayed for ***AnalystReadOnly roles for the levels below the Documents node These apply for *active* Contracts
No Flag used – its value is the STA_CD value from the documents corresponding DOC_STA table	DocStatus	Its value is determined by traversing up the rows

3.8.5. CollecitveEngineROVO

CollectiveEngineROVO objects related to engines which appear directly below a Qualification.
The data for the CollectiveEngineROVO object is collected from the following tables:
CALC_OPR → engine
CALC_OPR_TYP → engine type

The SQL query used for the CollectiveEngineROVO is:

```
select
        calc_opr.CPY_ID,
        calc_opr.BUS_ID,
        calc_opr.QUAL_ID,
        calc_opr.DOC_ID,
        calc_opr.CALC_OPR_DESC ENGINE_DESC,
        calc_opr.CALC_OPR_ID ID,
        calc_opr.PRNT_CALC_OPR_ID PARENT_ID,
        calc_opr.PRCDN_NUM PREC,
        calc_opr.DOC_CTNT_SUB_CD DOC_CONTENT_CODE,
        calc_opr_typ.CALC_OPR_CD ENGINE_CODE,
        calc_opr_typ.CALC_OPR_CLA_CD ENGINE_CLASS_CODE,
        calc_opr.STA_CD STATUS,
        calc_opr.RVS_CD REV,
        calc_opr.ORI_IX_CD ORIG,
        calc_opr.DES_IX_CD DEST,
        calc_opr.INCL_PRC_BID_FLG INCLUDE_PRICING_BID,
        calc_opr.INCL_RTE_GDE_FLG INCLUDE_PRIORITY_DURATION,
        calc_opr_typ.REQ_BAND_CD_FLG REQUIRE_BAND,
        calc_opr_typ.REQ_BAND_UOM_FLG REQUIRE_BAND_UOM,
        calc_opr_typ.REQ_TIER_CD_FLG REQUIRE_TIER,
        calc_opr_typ.REQ_ORI_ZONE_FLG REQUIRE_ORIGIN,
        calc_opr_typ.REQ_DES_ZONE_FLG REQUIRE_DEST,
        calc_opr.AUD_FLG,
        calc_opr_typ.CRE_CLC_FLG ADD_ENGINES_ABOVE,
        calc_opr_typ.CRE_BAS_FLG ADD_ENGINE,
        calc_opr_typ.CRE_OPRND_FLG ADD_DATA_GROUP
from
   SNIRATE.CALC_OPR calc_opr,
   SNIRATE.CALC_OPR_TYP calc_opr_typ
where
    calc_opr.CALC_OPR_TYP_ID = calc_opr_typ.CALC_OPR_TYP_ID
and calc_opr.PRNT_CALC_OPR_ID is NULL
```

For the CollectiveEngineROVO object, since it appears directly below the Qualifications, the PRNT_CALC_OPR_ID is **always null**.

The CollectiveEngineROVO view has no explicit bind variables.

The CollectiveEngineROVO uses the following View Accessors for three of its attributes

View Accessor Name	***Static LOV / SNI Foundation LOV***	***Attribute Applicable***	***App Code [SNI]***	***Domain Name [SNI]***	***Method in CollectiveEngineROVORowImpl to retrieve 'decoded' value***
CollectiveEngineRevisionVA	SNI Foundation LOV	Rev	RT	RATING_REVISION_CD	getRevValue()
CollectiveEngineStatusVA	SNI Foundation LOV	Status	RT	RATING_STATUS_CD	getStatusValue()
CollectiveEngineTypeVA	SNI Foundation LOV	EngineCode	RT	RATING_ENGINE	getEngineCodeValue()

For determining the display of the dynamic command links; the CollectiveEngineROVO has the following additional transient attributes:

Flag used from CALC_OPR_TYP or DOC_STA tables	Transient Variable [CollectiveEngineROVO]	Purpose
No Flag used – Value is determined by identifying the Contract row by traversing up the tree views using the **getContractStatus** method in the CollectiveEngine*ROVORowImpl* class	ContractStatus	Its value, either 'ACTV' or 'INAC' is determined by traversing up the rows.
No Flag used – Value is determined by identifying the Document row by traversing up the tree view using the ***getDocDataType*** method in the CollectiveEngineROVORowImpl class	DocDataType	To identify the document data type that the qualification belongs to
UPD_FLG from DOC_STA table – Value is determined by identifying the Document row by traversing up the tree view using the ***getIsDocEditiable***	IsDocEditable	To determine whether '***edit / view***' should be displayed for ***Analyst roles or whether '***audit / view***' link is displayed for ***AnalystReadOnly roles for the levels below the Documents node These apply for *active* Contracts
CRE_CLC_FLG from CALC_OPR_TYP table	IsAddEnginesAbove	To determine whether the '***add engine above***' link should be displayed for ***Analyst roles. These apply for *active* Contracts and when isDocEditable is true
CRE_BAS_FLG from CALC_OPR_TYP table	isAddEngines	To determine whether the '***add engine***' link should be displayed for ***Analyst roles. These apply for *active* Contracts and when isDocEditable is true
CRE_OPRND_FLG from CALC_OPR_TYP table	isAddDataGroups	To determine whether the '***add data group***' link should be displayed for ***Analyst roles. These apply for *active* Contracts and when isDocEditable is true
No Flag used – its value is the STA_CD value from the documents corresponding DOC_STA table	DocStatus	Its value is determined by traversing up the rows

3.8.6. BaseEngineROVO

IMPORTANT NOTE – There is a **departure from the nomenclature** used in the business terminology from the source code.

The distinguishable difference between objects from the CollectiveEngineROVO and the BaseEngineROVO is in the **PRNT_CALC_OPR_ID** (or the **PARENT_ID**) attribute.

Engine's ROVO	Distinguishing Attribute
CollectiveEngineROVO	PRNT_CALC_OPR_ID (or PARENT_ID) **IS NULL**
BaseEngineROVO	PRNT_CALC_OPR_ID (or PARENT_ID) **IS NOT NULL**

So CollectiveEngineROVO objects are those which ***immediately*** follow / or are ***just below*** QualificationROVO objects.

BaseEngineROVO objects have a non-null PRNT_CALC_OPR_ID (PARENT_ID) attribute the value of which is the CALC_OPR_ID (or ID) of a CollectiveEngineROVO.

In addition, BaseEngineROVO objects have a ***recursive relationship*** within themselves in that child or sub-BaseEngineROVO objects have their PRNT_CALC_OPR_ID (or PARENT_ID) equal to the CALC_OPR_ID (or ID) of another BaseEngineROVO

It is entirely possible for an object within the BaseEngineROVO domain to have an engine type where the ***CALC_OPR_CLA_CD*** (the engine classification code) is ***COLL*** – or Collective.

Only those BaseEngineROVO objects which **appear in the last level in the recursive hierarchy** of engines would be connected to an engine type where the ***CALC_OPR_CLA_CD*** is ***BASE***.

In addition for these particular Base Engines – those that appear in the lowest level of Engines, if they have an engine type where ***CRE_OPRND_FLG*** is 1 – then Data Groups are displayed for that Base Engine.

The data for the BaseEngineROVO is culled from the following tables:

- SNIRATE.**CALC_OPR**
- SNIRATE.**UOM_TYP**
- SNIRATE.**CALC_OPR_TYP**
- SNIRATE.**CTRC_REF**
- SNIRATE.**CTRC**

To enable ROVO object reuse some information for RateLookup functionality is also added into this query.

The SQL used for the BaseEngineROVO is:

```
SELECT
        calc_opr.CPY_ID,
        calc_opr.BUS_ID,
        calc_opr.DOC_ID,
        calc_opr.QUAL_ID,
        calc_opr.CALC_OPR_DESC ENGINE_DESC,
        calc_opr_typ.CRE_OPRND_FLG DISPLAY_DATAGROUP,
        calc_opr.CALC_OPR_ID ID,
        calc_opr.PRNT_CALC_OPR_ID PARENT_ID,
        calc_opr.PRCDN_NUM PREC,
        calc_opr.DOC_CTNT_SUB_CD DOC_CONTENT_CODE,
        calc_opr_typ.CALC_OPR_CD ENGINE_CODE,
        calc_opr_typ.CALC_OPR_CLA_CD ENGINE_CLASS_CODE,
        calc_opr.TIER_CD TIER,
        calc_opr.STA_CD STATUS,
        calc_opr.RVS_CD REV,
        calc_opr.ORI_IX_CD ORIG,
        calc_opr.STP_IX_CD,
        calc_opr.BAND_CD BAND_CODE,
        calc_opr.BAND_UOM_ID UOM_ID,
        calc_opr.LKUP_DAY_TYP_ID,
        calc_opr.INCL_RTE_GDE_FLG INCLUDE_PRIORITY_DURATION,
        calc_opr.ORI_CMR_ZONE_FLG,
        calc_opr.DES_CMR_ZONE_FLG,
        calc_opr.BAND2_CD,
        calc_opr.BAND2_UOM_ID,
        calc_opr.TRNST_CD TRANSIT_CODE,
        calc_opr.RTE_GDE_CD ,
        calc_opr.INCL_PRC_BID_FLG INCLUDE_PRICING_BID,
        calc_opr.MIN_RAT_AMT,
        calc_opr.MIN_RAT_UOM_ID,
        calc_opr.PBLCN_EXCL_IND,
        calc_opr.DES_IX_CD DEST,
        calc_opr_typ.REQ_BAND_CD_FLG REQUIRE_BAND_CODE,
        calc_opr_typ.REQ_BAND_UOM_FLG REQUIRE_BAND_UOM,
        calc_opr_typ.REQ_TIER_CD_FLG REQUIRE_TIER,
        calc_opr_typ.REQ_ORI_ZONE_FLG REQUIRE_ORIGIN,
        calc_opr_typ.REQ_DES_ZONE_FLG REQUIRE_DEST,
        calc_opr_typ.REQ_TRNST_CD_FLG REQUIRE_TRANSIT,
        uom_typ.UOM_DESC,
        calc_opr_typ.REQ_CTRC_REF_FLG REQUIRE_CONTRACT_REF,
        ctrc.CTRC_DESC CONTRACT_REF_DESC,
        calc_opr.AUD_FLG,
        calc_opr_typ.CRE_CLC_FLG ADD_ENGINES_ABOVE,
        calc_opr_typ.CRE_BAS_FLG ADD_ENGINE,
        calc_opr_typ.CRE_OPRND_FLG ADD_DATA_GROUP
FROM
   SNIRATE.CALC_OPR calc_opr,
   SNIRATE.UOM_TYP uom_typ,
   SNIRATE.CALC_OPR_TYP calc_opr_typ,
   SNIRATE.CTRC_REF ctrc_ref,
   SNIRATE.CTRC ctrc
WHERE
    calc_opr.CALC_OPR_ID = ctrc_ref.OBJ_ID(+)
and calc_opr.CALC_OPR_TYP_ID = calc_opr_typ.CALC_OPR_TYP_ID
and (:BindCalcOprId is null or calc_opr.calc_opr_id=:BindCalcOprId)
and calc_opr.PRNT_CALC_OPR_ID is NOT NULL
and calc_opr.BAND_UOM_ID = uom_typ.UOM_ID(+)
and ctrc_ref.ctrc_id = ctrc.ctrc_id (+)
```

The BaseEngineROVO has a single bind variable: BindCalcOprId.

The BaseEngineROVO uses the following View Accessors for its attributes

View Accessor Name	***Static LOV / SNI Foundation LOV***	***Attribute Applicable***	***App Code [SNI]***	***Domain Name [SNI]***	***Method in BaseEngineROVORowImpl to retrieve 'decoded' value***
BaseEngineRevisionVA	SNI Foundation LOV	Rev	RT	RATING_REVISION_CD	getRevValue()
BaseEngineStatusVA	SNI Foundation LOV	Status	RT	RATING_STATUS_CD	getStatusValue()
BaseEngineTypeVA	SNI Foundation LOV	EngineCode	RT	RATING_ENGINE	getEngineCodeValue()
BandTypeVA	SNI Foundation LOV	BandCode	RT	RATING_BAND_TYPE	getBandCodeValue()
EngineTierTypeLOVVA	Static LOV	Tier			getTierValue()
GeographyTypeVA	SNI Foundation LOV	Orig Dest	RT	GEOGRAPHY_TYPE	getOrigValue() getDestValue()
TransitTypeLOVVA	Static LOV	TransitCode			getTransitCodeValue()

Additional transient attributes used by the BaseEngineROVO include:

Flag used from CALC_OPR_TYP or DOC_STA tables	Transient Variable [BaseEngineROVO]	Purpose
No Flag used – Value is determined by identifying the Contract row by traversing up the tree views using the **getContractStatus** method in the BaseEngine*ROVORowImpl* class	ContractStatus	Its value, either 'ACTV' or 'INAC' is determined by traversing up the rows.
No Flag used – Value is determined by identifying the Document row by traversing up the tree view using the ***getDocDataType*** method in the BaseEngineROVORowImpl class	DocDataType	To identify the document data type that the qualification belongs to
No Flag used – but this information is displayed when the **BAND_CD** is **1** for the particular engine (from CALC_OPR)	BandInfo	If BAND_CD is 1 in the CALC_OPR table Then additional Band information is shown in the DisplayData for the Base Engine node
UPD_FLG from DOC_STA table – Value is determined by identifying the Document row by traversing up the tree view using the ***getIsDocEditiable***	IsDocEditable	To determine whether '***edit / view***' should be displayed for ***Analyst roles or whether '***audit / view***' link is displayed for ***AnalystReadOnly roles for the levels below the Documents node These apply for *active* Contracts
CRE_CLC_FLG from CALC_OPR_TYP table	IsAddEnginesAbove	To determine whether the '***add engine above***' link should be displayed for ***Analyst roles. These apply for *active* Contracts and when isDocEditable is true
CRE_BAS_FLG from CALC_OPR_TYP table	isAddEngines	To determine whether the '***add engine***' link should be displayed for ***Analyst roles. These apply for *active* Contracts and when isDocEditable is true
CRE_OPRND_FLG from CALC_OPR_TYP table	isAddDataGroups	To determine whether the '***add data group***' link should be displayed for ***Analyst roles. These apply for *active* Contracts and when isDocEditable is true
No Flag used – its value is the STA_CD value from the documents corresponding DOC_STA table	DocStatus	Its value is determined by traversing up the rows

3.8.7. DataGroupROVO

Data Groups form the last level of display in the Contract Tree – while technically there is a level below Data Groups – the *Rates* [taken from RTE table] – that *Rates* information is never visible in the Contract Tree.

The Data Groups information is extracted from the following tables:

- CALC_OPRND
- CALC_OPRND_GRP
- UOM_TYP

The SQL query used for DataGroups ROVO is:

```
select
  calc_oprnd_grp.CPY_ID,
  calc_oprnd_grp.BUS_ID,
  calc_oprnd_grp.DOC_ID,
  uom_typ.UOM_DESC DESCRIPTION,
  calc_oprnd_grp.CRCY_CD CURRENCY,
  calc_oprnd_grp.CALC_OPR_ID,
  calc_oprnd_grp.CALC_OPRND_GRP_ID,
  calc_oprnd_grp.STA_CD STATUS,
  calc_oprnd_grp.RVS_CD REV,
  count(calc_oprnd.calc_oprnd_id)
from
    SNIRATE.UOM_TYP uom_typ,
    SNIRATE.CALC_OPRND_GRP calc_oprnd_grp,
    SNIRATE.CALC_OPRND calc_oprnd
where
    calc_oprnd.calc_oprnd_grp_id(+) = calc_oprnd_grp.CALC_OPRND_GRP_ID
and uom_typ.uom_id = calc_oprnd_grp.uom_id
group by
  calc_oprnd_grp.CPY_ID,
  calc_oprnd_grp.BUS_ID,
  calc_oprnd_grp.DOC_ID,
  uom_typ.UOM_DESC,
  calc_oprnd_grp.CRCY_CD,
  calc_oprnd_grp.CALC_OPR_ID,
  calc_oprnd_grp.CALC_OPRND_GRP_ID,
  calc_oprnd_grp.STA_CD,
  calc_oprnd_grp.RVS_CD
```

The DataGroupROVO uses no explicit bind variable.

The DataGroupROVO uses the following View Accessors for its attributes

View Accessor Name	***Static LOV / SNI Foundation LOV***	***Attribute Applicable***	***App Code [SNI]***	***Domain Name [SNI]***	***Method in*** DataGroup***ROVORowImpl to retrieve 'decoded' value***
DataGroupRevisionVA	SNI Foundation LOV	Rev	RT	RATING_REVISION_CD	getRevValue()
DataGroupStatusVA	SNI Foundation LOV	Status	RT	RATING_STATUS_CD	getStatusValue()

Additional transient variables used by the DataGroupROVO include:

Flag used from CALC_OPR_TYP or DOC_STA tables	Transient Variable [DataGroupROVO]	Purpose
No Flag used – Value is determined by identifying the Contract row by traversing up the tree views using the **getContractStatus** method in the DataGroup*ROVORowImpl* class	ContractStatus	Its value, either 'ACTV' or 'INAC' is determined by traversing up the rows.
No Flag used – Value is determined by identifying the Document row by traversing up the tree view using the ***getDocDataType*** method in the DataGroupROVORowImpl class	DocDataType	To identify the document data type that the qualification belongs to
UPD_FLG from DOC_STA table – Value is determined by identifying the Document row by traversing up the tree view using the ***getIsDocEditiable***	IsDocEditable	To determine whether '***edit / view***' should be displayed for ***Analyst roles or whether '***audit / view***' link is displayed for ***AnalystReadOnly roles for the levels below the Documents node These apply for *active* Contracts
No Flag used – its value is the count of Data Group records from the CALC_OPRND table **count(calc_oprnd.calc_oprnd_id)**	Records	To display the number of DataGroup records
No Flag used – its value is the STA_CD value from the documents corresponding DOC_STA table	DocStatus	Its value is determined by traversing up the rows

3.9. Technique used for extracting LOV information

An LOV is essentially a front-end ADF UI component used for selecting items from lists or drop-down components. Since *java.lang.String* objects are used to render the information at the DisplayLabel and DisplayData for each facet in the tree, we need to perform additional steps in the ROVORowImpl classes to extract the decoded information.

Aliases are often used within the contents of the SQL query in ROVOs to extract the data from the database. In addition the values retrieved from the database are often in a coded form, for which LOVs act as a mechanism to gather the actual logical contents of the coded values (a sort of 'decoding' if you may).

Depending on whether the attribute applies to a static LOV from the ***sni.irct.model.queries.lov*** package or is a subclass of the ***sni.foundation.lovs.quries.SNIListOfValuesVO*** class; the technique used to gather the LOV related information is different.

3.9.1. Static LOVs from the sni.irct.model.queries.lov package

For static LOVs; those which are view accessors from the ***sni.irct.model.queries.lov*** package; the following technique is used within the ***ROVORowImpl classes:

[We would need to inspect the particular LOV in question to determine which its key attribute is]

- We first get the coded value from the attribute
- Next we call upon the LOV View Accessor and execute its associated query
- We iterate through the LOV values and when a match is found, return the value associated for that code

```
private String getTierValue()
  {
    String tierValue = null;
    if (showTierData())
    {
      // NOTE --> EngineTierTypeLOV is a static LOV
      String tier = getTier();        → encoded value of the attribute existing in the ROVO
      RowSet tierRowSet = getEngineTierTypeLOVVA();
      RowSetIterator niter = tierRowSet.createRowSetIterator(null);
      while (niter.hasNext())
      {
        EngineTierTypeLOVRowImpl tierLOVRow =
          (EngineTierTypeLOVRowImpl) niter.next();
        if (tierLOVRow.getTierCd().equalsIgnoreCase(tier))   → find the match for the coded LOV
        {
          tierValue = tierLOVRow.getTierDesc();  → return the associated value
          break;
        }
      }
      niter.closeRowSetIterator();
    }
    return tierValue;
  }
```

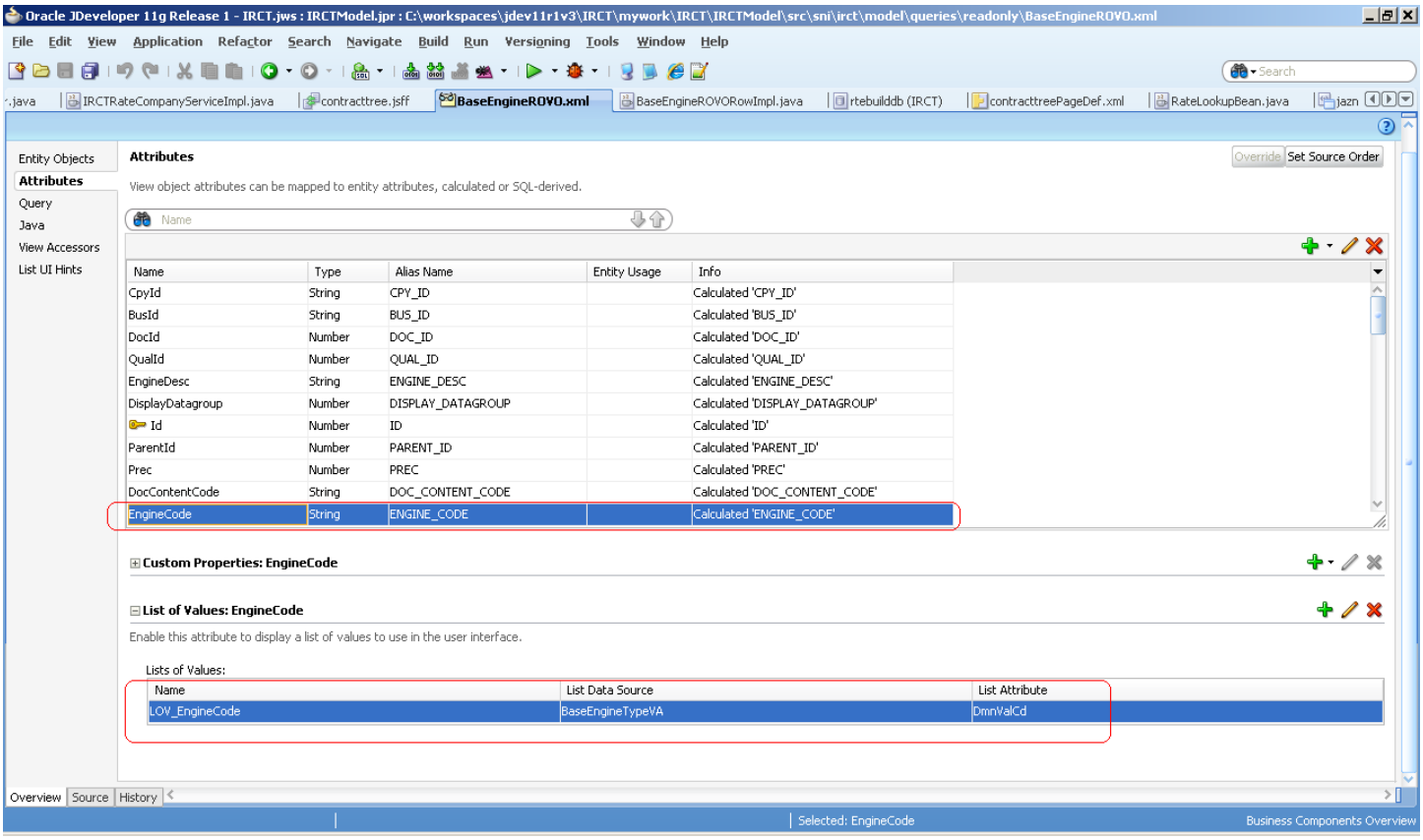

The following table describes which Contract Tree ROVO objects use LOVs from the sni.irct.model.queries.lov package

Contract Tree ROVO	Attribute to which LOV is applied	Attribute in RowImpl which returns value	View Accessor LOV used from sni.irct.model.queries.lov package	LOV Key Attribute	LOV Associated Value used
DocumentROVO	CurrencyFrequency	CurrencyFrequencyValue	CurrencyFrequencyLOV	CurrencyFrequencyCd	CurrencyFrequencyDesc
	DistPack	DistPackValue	DistancePkgLOV	GeoPkgCd	LegNam
	Usage	UsageValue	UsageLOV	UsageCd	UsageDesc
BaseEngineROVO	Tier	TierValue	EngineTierTypeLOV	TierCd	TierDesc
	TransitCode	TransitCodeValue	TransitTypeLOV	TrnstCd	TrnstDesc

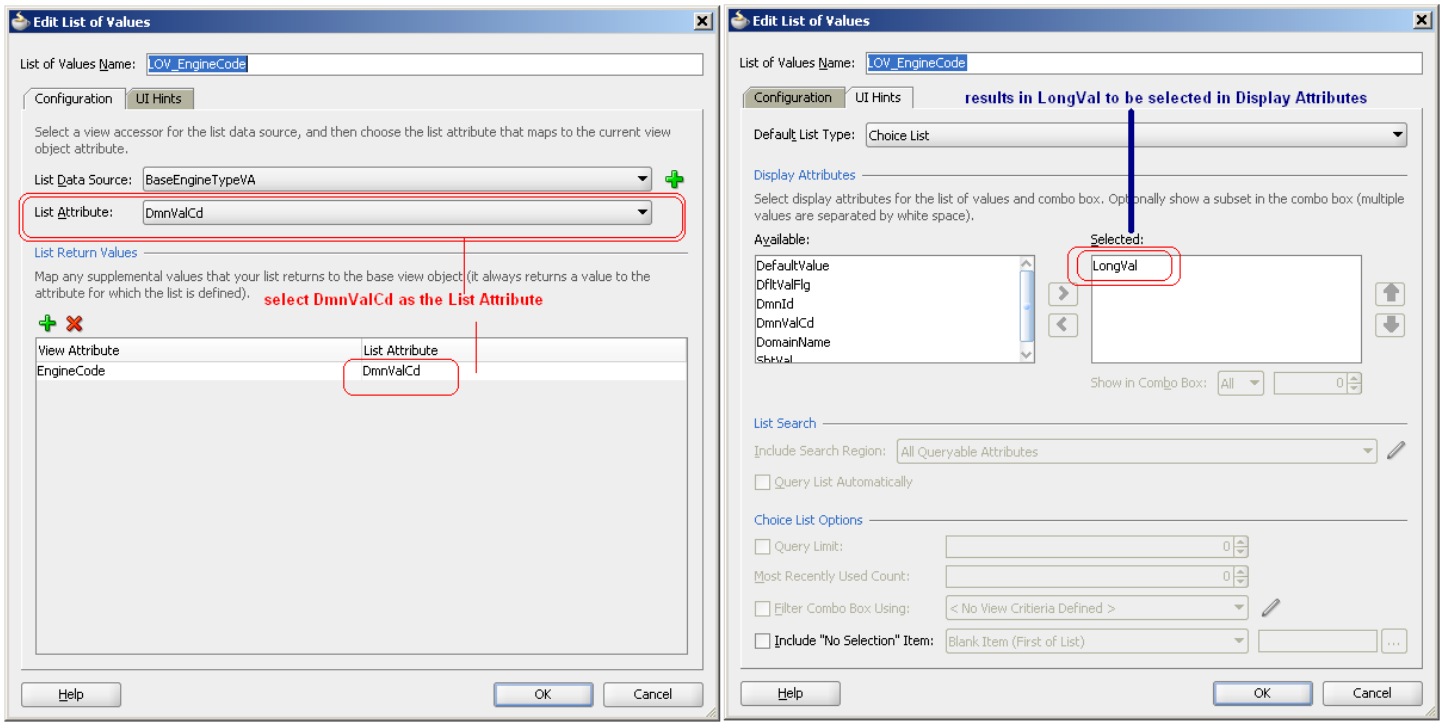
3.9.2. SNI Foundation LOVs subclassing sni.foundation.lovs.quries.SNIListOfValuesVO

Note that for these attributes within the *List Of Values* section of the *Attributes* tab; the List Attribute to be selected is ***always*** **DmnValCd.** By selecting this default domain value code; within the Configurations section of the '*Edit List of Values*' popup – for the UI Hints '*LongVal*' will always be *preselected.*

The images below display the example for EngineCode within the BaseEngineROVO class



LOV info related to the EngineCode attribute



For dynamic LOVs which sub class the sni.foundation.lovs.quries.SNIListOfValuesVO class; the following technique is used within the ROVORowImpl classes:

-- getEngineCodeValue() method from BaseEngineROVOROWImpl class --

```
private String getEngineCodeValue()
  {
    String type = getEngineCode();      → encoded value of the attribute existing in the ROVO
    String typeValue = null;
    RowSet typeRowSet = getBaseEngineTypeVA();
    if (typeRowSet != null)
    {
      typeRowSet.reset();
      if (typeRowSet != null && StringUtils.isNotBlank(type))
      {
        typeValue = SNIListOfValuesVOImpl.getLongValue(typeRowSet, type);
      }
      typeRowSet.closeRowSet();
    }
    return typeValue;      → return the associated value
  }
```


3.10. Contract Tree View Links

The view links form the join condition between the levels in the tree.
The view links exist in the ***sni.irct.model.queries.viewlinks*** package.
Note that the corresponding views connected to are existing within the ***sni.irct.model.queries.readonly*** package.

As we expand or collapse the nodes in the tree we are activating the view links to execute the queries in the next level and pass on bind parameters. All the view links posses ***1 to **** (1 to Many) as we traverse from Parent (Source) to Child (Destination) views.

The following table displays the View Links, the bind parameters involved and methods (called from the Row implementation classes)

View Link	Source View	Source BindParam	Destination View	Destination BindParam	ParentRow Accessor : return type	DestinationRow Accessor : return type
CtrcForRatCpyVL	RatingCompanyROVO	Id	ContractROVO	CpyId	*RatingCompanyROVORowImpl.* getContracts : RowIterator	*ContractROVORowImpl.* getRatingCompany : Row
DocForCtrcVL	ContractROVO	Id	DocumentROVO	CtrcId	*ContractROVORowImpl.* getDocuments : RowIterator	*DocumentROVORowImpl.* getContract : Row
QualForDocVL	DocumentROVO	Id	QualificationROVO	DocId	*DocumentROVORowImpl.* getQualifications : RowIterator	*QualificationROVORowImpl.* getDocument : Row
CollectiveEngForQualVL	QualificationROVO	Id	CollectiveEngineROVO	QualId	*QualificationROVORowImpl.* getCollectiveEngines : RowIterator	*CollectiveEngineROVORowImpl.* getQualification : Row
BaseEngForCollectiveEngVL	CollectiveEngineROVO	Id	BaseEngineROVO	ParentId	*CollectiveEngineROVORowImpl.* getBaseEngines : RowIterator	*BaseEngineROVORowImpl.* getCollectiveEngine : Row
BaseEngForBaseEngVL	BaseEngineROVO	Id	BaseEngineROVO	ParentId	*BaseEngineROVORowImpl.* getSubBaseEngines : RowIterator	*BaseEngineROVORowImpl.* getParentBaseEngine : Row
DatagrpForBaseEngVL	BaseEngineROVO	Id	DataGroupROVO	CalcOprId	*BaseEngineROVORowImpl.* getDataGroups : RowIterator	*DataGroupROVORowImpl.* getBaseEngine : Row

Note – Since we are already executing the RatingCompanyROVO with business id as a bind variable – an additional bind variable of business id does not exist for the CtrcForRatCpyVL view link

3.11. The ContractTree UI

The User Interface for displaying and rendering the Contract Tree are dependent on the following artifacts:

Artifact		***Directory Location***
maintaincontractshome.xml	ADF Task Flow	\IRCTUI\public_html\taskflows\rating
contractmaintenance.jspx	JSF Page	\IRCTUI\public_html\jsps
contractmaintenancePageDef.xml	Page Definition	\IRCTUI\adfmsrc\jsps
contractmaintenance-config	ADF Task Flow	\IRCTUI\public_html\taskflows\rating
contractmaintenancePageDef.xml	Page Definition	\IRCTUI\adfmsrc\jsff\rating
contractmaintenance.jsff	JSF Fragment	\IRCTUI\public_html\jsff\rating
contractmaintenancePageDef.xml	Page Definition	\IRCTUI\adfmsrc\jsff\rating
contractTree.jsff	JSF Fragment	\IRCTUI\public_html\jsff\rating
contractTreePageDef. xml	Page Definition	\IRCTUI\adfmsrc\jsff\rating

The ***contractmaintenance.jspx*** page is the JSF page which holds the ***maintaincontractshome*** task flow as a dynamic region.
The ***maintaincontractshome*** holds a single view – ***contractmaintenance.jsff*** – which is its default activity.
The ***contractmaintenance.jsff*** holds an ADF *navigationPane* component. This *navigationPane component* has three *commandNavigationItems* – for "Maintain Contracts" [which holds the UI for the Contract tree and its related screens], ""Processing Status" and "Publication Processing Status". If a User is granted access to the Contract Tree View, the navigationPane is set to display the "Maintain Contracts" tab by default – this way when one enters or leaves the task flow one goes straight to the region where the Contract Tree is rendered. This ensures that if the user were to click on the related tabs of "Processing Status" or "Processing Pubication Status" and then perform a "Rate Lookup" and then click on the dynamic link on that screen to see the Contract Tree – the user would indeed be directed to the screen displaying the Contract Tree, instead of the last visible tab clicked.
The tabs of 'Maintain Contracts', 'Processing Status' and 'Publication Processing Status' are displayed based on the User's Security Roles.

Code extract from contractmaintenance.jsff

```
. . .
<af:navigationPane id="pt_np1">
<!-- Maintain Contracts CANNOT be viewed by RSPublicationAnalyst and/or RSPublicationInitiator -->
  <af:commandNavigationItem text="#{irctuiBundle.MAINTAIN_CONTRACTS}" id="cni1" immediate="true"
    rendered="#{securityContext.userInRole['RSRateAnalyst, RSRateAnalystReadOnly, RSTransitAnalyst,
                  RSTransitAnalystReadOnly, RSRoutingGuideAnalyst, RSRoutingGuideAnalystReadOnly']}"
    selected="#{(securityContext.userInRole['RSRateAnalyst, RSRateAnalystReadOnly, RSTransitAnalyst,
                  RSTransitAnalystReadOnly, RSRoutingGuideAnalyst, RSRoutingGuideAnalystReadOnly'] and
                    (empty viewScope.selectedTab)) or viewScope.selectedTab==1}">
      <af:setActionListener from="1" to="#{viewScope.selectedTab}"/>
  </af:commandNavigationItem>
  <af:commandNavigationItem text="#{irctuiBundle.PROCESSING_STATUS}" id="cni2" immediate="true"
    rendered="#{securityContext.userInRole['RSPublicationAnalyst, RSPublicationInitiator']}"
    selected="#{viewScope.selectedTab==2 or (!securityContext.userInRole['RSRateAnalyst,
            RSRateAnalystReadOnly, RSTransitAnalyst, RSTransitAnalystReadOnly, RSRoutingGuideAnalyst,
            RSRoutingGuideAnalystReadOnly'] and (empty viewScope.selectedTab))}">
      <af:setActionListener from="2" to="#{viewScope.selectedTab}"/>
  </af:commandNavigationItem>
  <af:commandNavigationItem text="#{irctuiBundle.PUBLICATION_PROCESSING_STATUS}" id="cni3" immediate="true"
     rendered="#{securityContext.userInRole['RSPublicationAnalyst, RSPublicationInitiator']}"
     selected="#{viewScope.selectedTab==3}">
       <af:setActionListener from="3" to="#{viewScope.selectedTab}"/>
  </af:commandNavigationItem>
</af:navigationPane>
. . .
   <af:group>
     <af:region value="#{bindings.contractmaintenanceconfig1.regionModel}"
       rendered="#{(securityContext.userInRole['RSRateAnalyst, RSRateAnalystReadOnly, RSTransitAnalyst,
                     RSTransitAnalystReadOnly, RSRoutingGuideAnalyst, RSRoutingGuideAnalystReadOnly'] and
                     ( empty viewScope.selectedTab)) or viewScope.selectedTab==1}"
                                                  id="r1"/>
. . .
```

3.11.1. The Contract Maintenance Task Flow with the Contract Tree in Focus

The ***contractmaintenance-config*** taskflow is a bounded task flow and the main parent taskflow for displaying the contract tree.

The central screen within the taskflow is contracttree.jsff
The default activity – the first activity to be invoked upon entry – for the task flow is ***setUpRatingCompanyForUser***

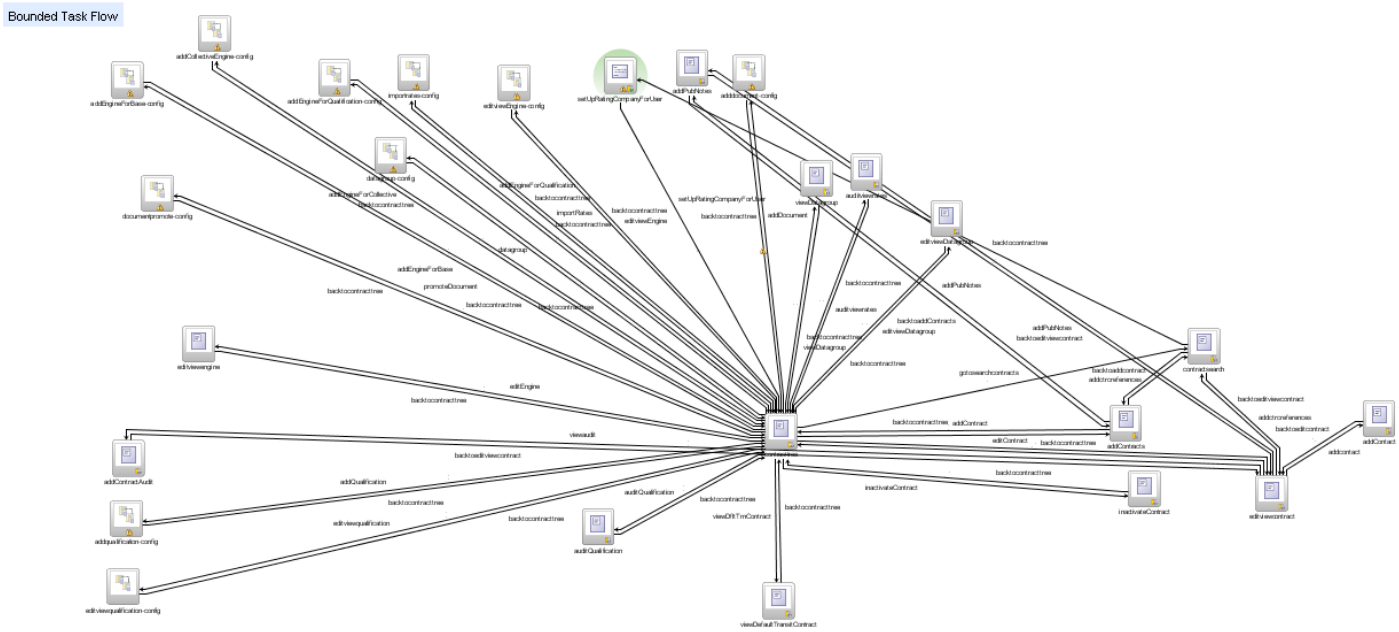
Dynamic Command Links and buttons on the contracttree.jsff screen direct the user to leading screens or perform certain predefined activities.

There is a **hybrid** approach followed with some leading screens being accessed directly through control-flow activities and some screens via child task flows.

The ***contractmaintenance-config*** taskflow takes **two input parameters**

Input Parameter	Value	Type	Scope
sourcePath	#{sessionScope.sourcePath}	java.lang.String	Session
lkprteTaskFlowParamMap	#{sessionScope.lkprteTaskFlowParamMap}	java.util.Map	Session

It is expected that future avatars of the Rate Server Platform we will have child task flows for ALL the leading screens. This will make the architecture more streamlined and homogenous and not appear as messy as it does in the contractmaintenance-config diagram.



The central screen is ***contracttree.jsff***; which is used to render the Contract Tree.

The first [default] activity upon entering the contractmainetence-config task flow is the ***setUpRatingCompanyForUser*** activity. When activated it executes the ***setUpRatingCompanyForUser*** method within the IRCTRateCompanyServiceImpl class.

Leading screens or task flows are each accessed by uniquely identifying activities.
All of them use the activity ***backtocontracttree*** to return to the contracttree.jsff screen.

The contract maintenance task flow has two managed beans:
ContractTreeHandler – In pageFlowScope, controls interactions with the Contract Tree
ContractSearchBean – In pageFlowScope, controls interactions with the Contract Search

The contractmaintenace-config task flow is of type ***Train*** to enable the user to go to specific screens.

The task flow enables reentry → Task Flow Reentry type is ***reentry-allowed***

The **finalizer** for the task flow is the ***cleanup*** method in the ContractTreeHandler class.
Its purpose is to remove the session scoped variables of sourcePath and lkprteTaskFlowParamMap when the user exits the task flow

The activities and their related views and task flows for the contrractmaintence-config task flow are displayed below:

Activities

Activity Type	Name *
view	contracttree
view	contractsearch
task-flow-call	editviewqualification-config
view	addContracts
view	auditQualification
view	editviewcontract
view	inactivateContract
view	editviewengine
view	auditviewrates
view	editviewDatagroup
view	viewDatagroup
task-flow-call	addEngineForQualification-config
task-flow-call	addEngineForBase-config
task-flow-call	datagroup-config
view	addPubNotes
method-call	setUpRatingCompanyForUser
task-flow-call	editviewEngine-config
task-flow-call	adddocument-config
task-flow-call	importrates-config
view	addContractAudit
view	addContact
task-flow-call	addqualification-config
view	viewDefaultTransitContract
task-flow-call	addCollectiveEngine-config
task-flow-call	documentpromote-config

3.11.2. The ADF Tree implementation of the Contract Tree

The ADF Tree component is used to display the Contract Tree.
It is located within the ***contracttree.jsff*** screen

```
<af:tree value="#{bindings.RatingCompanyROVO.treeModel}"
    var="node" rowSelection="single" id="t1"
    expandAllEnabled="true"
    summary="Contract Tree with model developed declaratively"
    contentDelivery="immediate"
    binding="#{pageFlowScope.ContractTreeHandler.contractTree}"
    displayRow="selected"
    selectionListener="#{pageFlowScope.ContractTreeHandler.getSelection}"
    rowDisclosureListener="#{pageFlowScope.ContractTreeHandler.handleRowDisclosure}"
    partialTriggers="::pasteWith ::pasteWithout">
  <f:facet name="nodeStamp">
    <h:panelGroup id="pg2">
      <af:outputLabel value="#{node.DisplayLabel}" id="ol1"/>
      .... Command Links ...
      <h:panelGrid id="pg1">
          <af:outputText value="#{node.DisplayData}" id="ot1"/>
           </h:panelGrid>
    </h:panelGroup>
  </f:facet>
</af:tree>
```

Attributes of af:tree

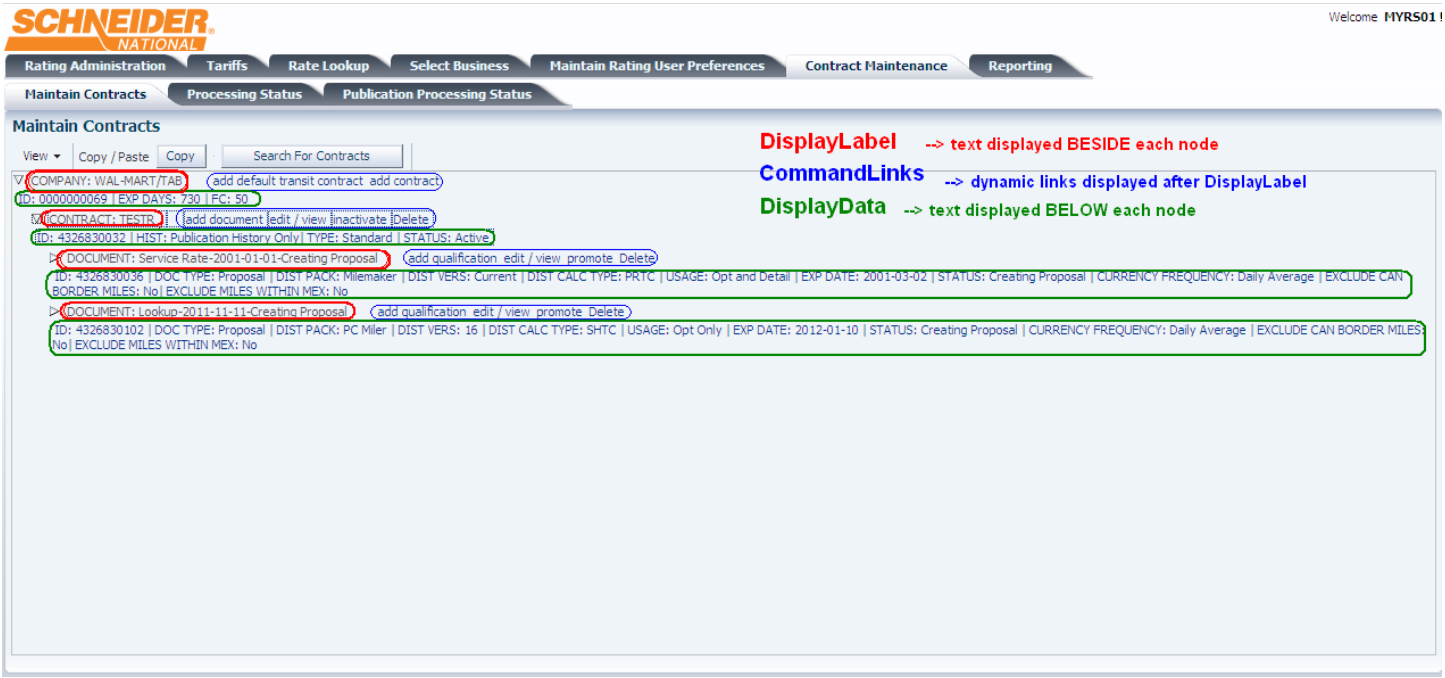
value="#{bindings.RatingCompanyROVO.treeModel}"	The hierarchy for the Contract Tree data is bound to the iterators originating from the RatingCompanyROVO.
contentDelivery="immediate"	Enables lazy loading of the data
binding="#{pageFlowScope.ContractTreeHandler.contractTree}"	The ***contractTree*** attribute in the ***ContractTreeHandler*** class stores the component instance of the model. The attribute is of type oracle.adf.view.rich.component.rich.data.***RichTree***
selectionListener="#{pageFlowScope.ContractTreeHandler.getSelection}"	This method helps identify which nodes are selected – it is used in the Copy / Paste functionality
rowDisclosureListener= "#{pageFlowScope.ContractTreeHandler.handleRowDisclosure}"	This method is triggered to handle expansions and collapses of nodes in the Contract Tree. It calls upon the *expandTree* method when the tree is displayed via a Rate Lookup
partialTriggers="::pasteWith ::pasteWithout"	The IDs of the Paste buttons trigger partial updates of the tree – without a complete page refresh
var="node"	The name of the reference for each level in the Contract Tree
expandAllEnabled="true"	Renders menu items for expanding and collapsing the nodes once a selection is made

3 levels of information are **always** displayed ***for each node*** in the Contract Tree.

1- DisplayLabel
2- Dynamic CommandLinks
3- DisplayData

Each level in the tree has a unique attribute to help determine which set of CommandLinks to display

Level in the Contract Tree	***Uniquely Distinguishing Attribute***
Company	Name not null and Hist is null
Contract	Name is null and Hist is not null
Document	DocType is not null
Qualification	QualDesc is not null
CollectiveEngine	EngDesc is not null and ParentID is null
BaseEngine	EngDesc is not null and ParentID is not null
DataGroup	Currency is not null



3.11.3. The ADF Tree Bindings for the Contract Tree

The \IRCT\mywork\IRCT\IRCTUI\adfmsrc\jsff\rating**contracttreePageDef.xml** holds the information for the databound components for the Contract Tree.

NOTE – Please DO NOT, repeat, DO NOT edit the bindings for the <tree using the GUI editor – if you click on the binding for the RatingCompanyROVO – and wish to view it, go ahead – but avoid clicking on 'OK' – click on 'Cancel' instead.

All changes within the *<tree>* binding for *RatingCompanyROVO* MUST be Hand-Coded in the *contracttreePageDef.xml* file

The RatingCompanyROVO is bound as an iterator to the ContractTree.

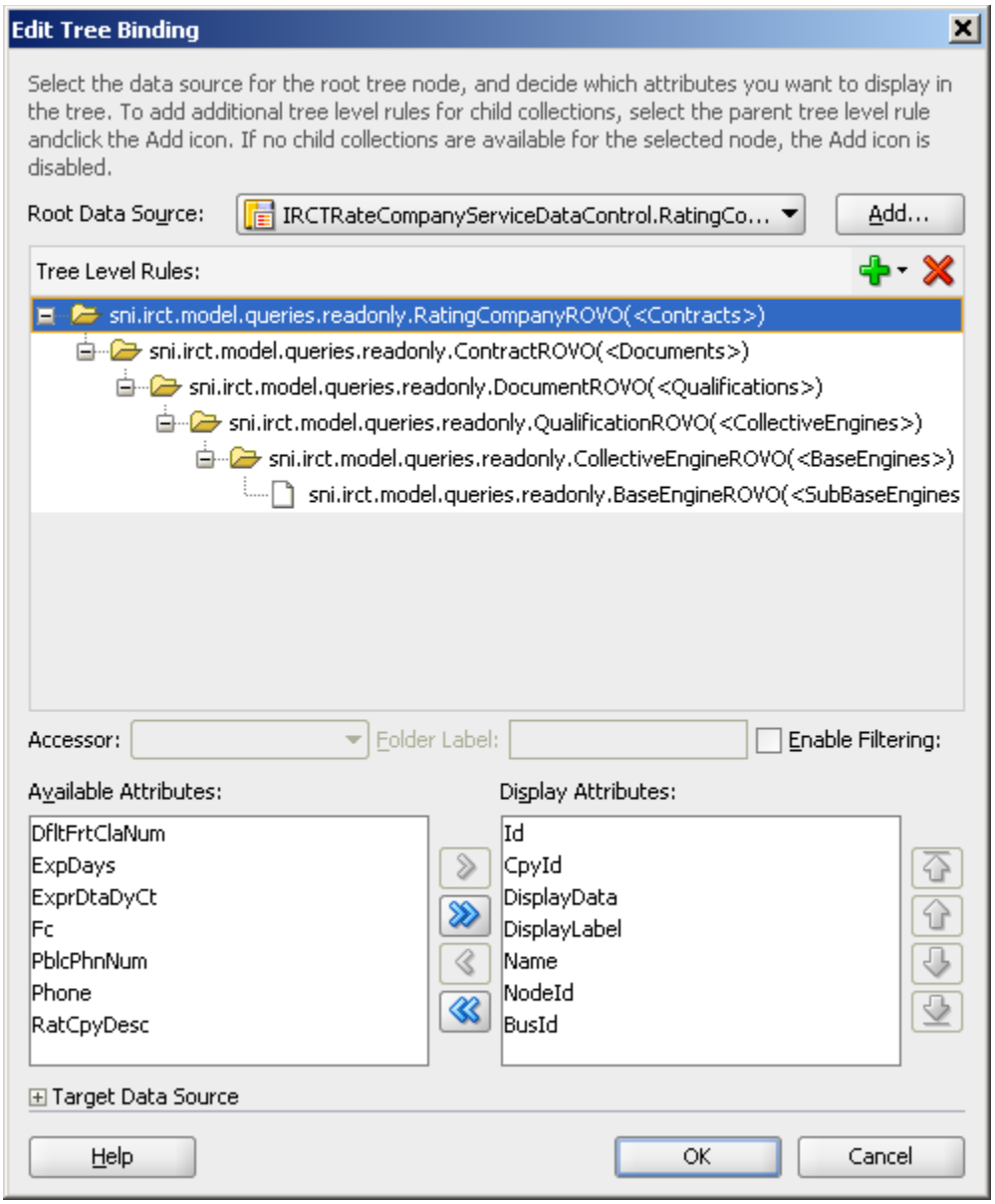
```
<executables>
<iterator Binds="RatingCompanyROVO" RangeSize="25"
          DataControl="IRCTRateCompanyServiceDataControl"
          id="RatingCompanyROVOIterator" CacheResults="false"/>
```

Note – CacheResults is set to false. This ensures that changes made as we go back and forth to the Contract Search – or changes made within the contractmaintenance-config task flow – the tree will be refreshed.

The <af:tree> component is then bound to the above RatingCompanyROVO iterator.
The tree iterBinding holds a collection of **nodeDefinition**s – each nodeDefinition for a node level in the tree.

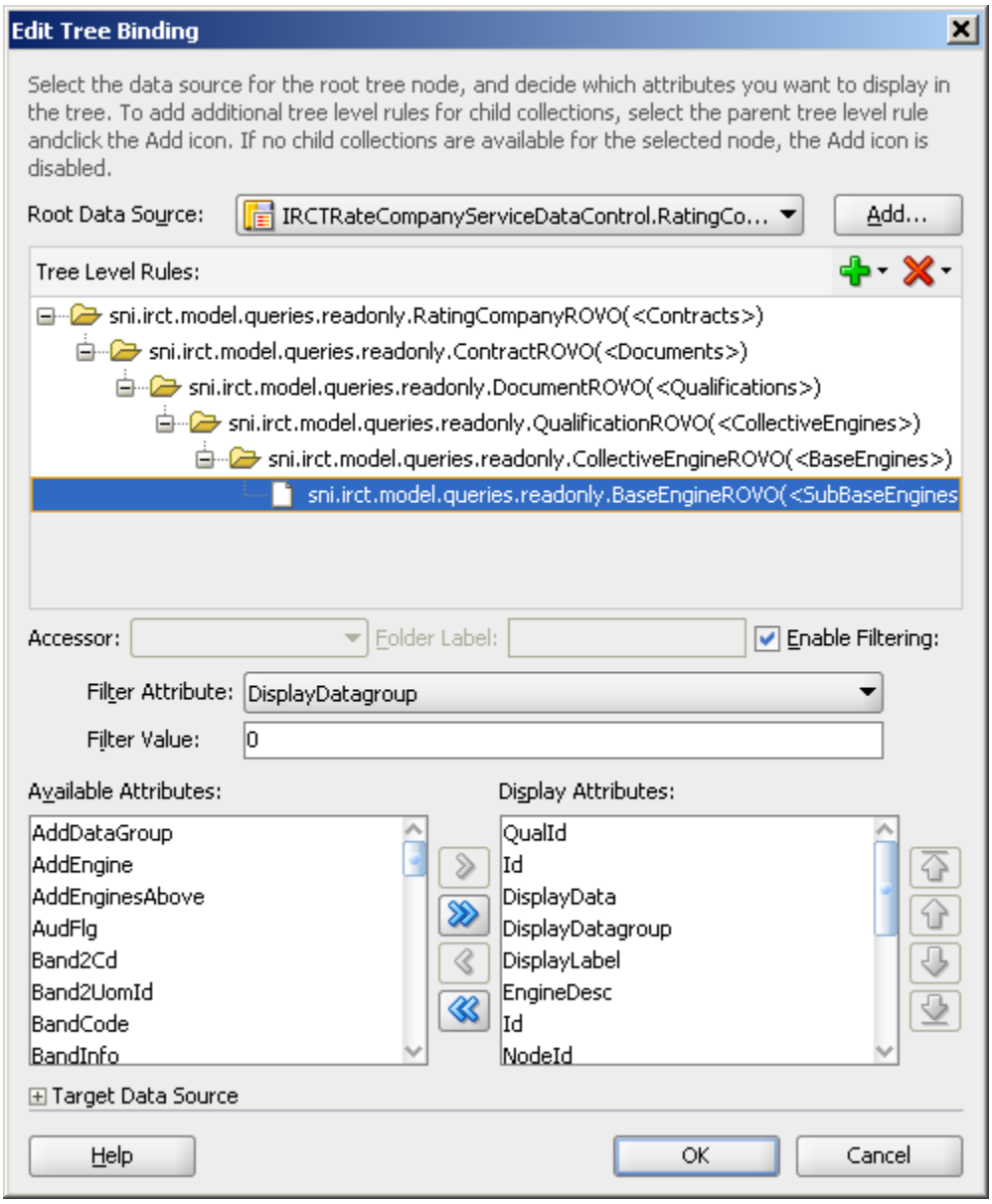
The <af:tree> is bound to the RatingCompanyROVO iterator with a collection of nodeDefinitions as follows:

```
<tree IterBinding="RatingCompanyROVOIterator" id="RatingCompanyROVO">
   <nodeDefinition DefName="sni.irct.model.queries.readonly.RatingCompanyROVO" Name="RatingCompanyROVO0">…
   <nodeDefinition DefName="sni.irct.model.queries.readonly.ContractROVO" Name="RatingCompanyROVO1">…
   <nodeDefinition DefName="sni.irct.model.queries.readonly.DocumentROVO" Name="RatingCompanyROVO2">…
   <nodeDefinition DefName="sni.irct.model.queries.readonly.QualificationROVO" Name="RatingCompanyROVO3">…
   <nodeDefinition DefName="sni.irct.model.queries.readonly.CollectiveEngineROVO" Name="RatingCompanyROVO4">...
  <nodeDefinition DefName="sni.irct.model.queries.readonly.BaseEngineROVO"
              DiscrName="DisplayDatagroup" DiscrValue="0"
              Name="RatingCompanyROVO5">…
  <nodeDefinition DefName="sni.irct.model.queries.readonly.BaseEngineROVO"
              DiscrName="DisplayDatagroup" DiscrValue="1"
              Name="RatingCompanyROVO55">…
<nodeDefinition DefName="sni.irct.model.queries.readonly.DataGroupROVO"
              Name="RatingCompanyROVO6">…
</tree>
```

The above images displays the bindings for the adf:tree component. The top lelvel view used is the RatingCompanyROVO view.
As mentioned previously:, for each level the Display Attributes for ‘Display Data’, ‘Id’, ‘NodeId’ and ‘DisplayLabel’ are always selected.
For determining which of the dynamic command links need to be displayed as per the level additional attributes are selected.

Note that the in image we only choose to display the tree in the GUI ***upto*** to first top level of BaseEngines:



The levels below - **namely the sub levels of BaseEngines and the DataGroups** - are **hand coded** in the \IRCT\mywork\IRCT\IRCTUI\adfmsrc\jsff\rating**contracttreePageDef.xml** file.

Depending on the level on the Contract Tree and what conditions may be used to render the dynamic command links for the particular level – certain specific 'Display Attributes' are selected for each ROVO utilized by the Contract Tree

ROVO – nodeDefinition level	Display Attributes in contracttreePageDef.xml	Accessor name for the node below
RatingCompanyROVO - Company	Id, CpyId, DisplayData, DisplayLabel, Name, NodeId, BusId,	Contracts
ContractROVO - Contracts	CpyId DisplayData BusId DisplayLabel Hist Id NodeId Status Name	Documents
DocumentROVO - Documents	CtrcId DisplayData DisplayLabel DocType CanCreateQuals IsUpdatable IsDeletable Id IsPromotable NodeId ContractStatus DocContentCode IsEditable	Qualificatons
QualificationROVO - Qualifications	DocId DisplayData DisplayLabel Id NodeId QualDesc ContractStatus DocDataType IsDocEditable DocStatus	CollectiveEngines
CollectiveEngineROVO – CollectiveEngines	QualId DisplayData DisplayLabel EngineDesc Id NodeId ParentId ContractStatus IsAddEnginesAbove	BaseEngines

	IsAddEngines IsAddDataGroups DocDataType IsDocEditable DocStatus	
BaseEngineROVO – [Parent Base Engines] **DiscrName="DisplayDatagroup"** **DiscrValue="0"**	QualId Id DisplayData DisplayDatagroup DisplayLabel EngineDesc NodeId ParentId ContractStatus IsAddEnginesAbove IsAddEngines IsAddDataGroups DocDataType IsDocEditable DocStatus	SubBaseEngines
BaseEngineROVO – [Last Base Engine] **DiscrName="DisplayDatagroup"** **DiscrValue="1"**	QualId Id DisplayData DisplayDatagroup DisplayLabel EngineDesc NodeId ParentId ContractStatus IsAddEnginesAbove IsAddEngines IsAddDataGroups DocDataType IsDocEditable DocStatus	DataGroups
DataGroupROVO – Data groups	Currency CalcOprndGrpId DisplayData DisplayLabel NodeId ContractStatus DocDataType IsDocEditable DocStatus	

3.11.4. The Hack for SubBaseEngines and DataGroups

This setup cannot be done using the GUI code editor – it has to be done manually.
Basically we are setting it up such that the engine hierarchy (for 3 layers of engines) displayed is as follows:

```
Qualifications
  |
  |__ CollectiveEngines [or records from CALC_OPR where PRNT_CALC_OPR_ID is null]
          |
          |__BaseEngines [or records from CALC_OPR where PRNT_CALC_OPR_ID is not null
               |                          and CRE_OPRND_FLG is 0
               |              Note → for these records CALC_OPR_CLA_CD may be 'COLL', 'BASE', or 'DPND']
               |
               |__ (Sub) BaseEngines [or records from CALC_OPR where PRNT_CALC_OPR_ID is not null
                          |                          and CRE_OPRND_FLG is 1
                          |          Note → for these records CALC_OPR_CLA_CD is ALWAYS 'BASE'] |
                          |
                          |_ DataGroups
```

The following are the nodeDefinition contents of the ***contractTreePageDef.xml*** file starting from the Qualifications level:

```
<nodeDefinition DefName="sni.irct.model.queries.readonly.QualificationROVO"
                    Name="RatingCompanyROVO3">
     <AttrNames>
       <Item Value="DocId"/>
       <Item Value="DisplayData"/>
       <Item Value="DisplayLabel"/>
       <Item Value="Id"/>
       <Item Value="NodeId"/>
       <Item Value="QualDesc"/>
       <Item Value="ContractStatus"/>
       <Item Value="DocDataType"/>
       <Item Value="IsDocEditable"/>
       <Item Value="DocStatus"/>
     </AttrNames>
     <Accessors>
       <Item Value="CollectiveEngines"/>
     </Accessors>
   </nodeDefinition>
   <nodeDefinition DefName="sni.irct.model.queries.readonly.CollectiveEngineROVO"
                    Name="RatingCompanyROVO4">
     <AttrNames>
       <Item Value="QualId"/>
       <Item Value="DisplayData"/>
       <Item Value="DisplayLabel"/>
       <Item Value="EngineDesc"/>
```



```
      <Item Value="Id"/>
      <Item Value="NodeId"/>
      <Item Value="ParentId"/>
      <Item Value="ContractStatus"/>
      <Item Value="IsAddEnginesAbove"/>
      <Item Value="IsAddEngines"/>
      <Item Value="IsAddDataGroups"/>
      <Item Value="DocDataType"/>
      <Item Value="IsDocEditable"/>
      <Item Value="DocStatus"/>
    </AttrNames>
    <Accessors>
      <Item Value="BaseEngines"/>
    </Accessors>
  </nodeDefinition>
  <nodeDefinition DefName="sni.irct.model.queries.readonly.BaseEngineROVO"
                  DiscrName="DisplayDatagroup" DiscrValue="0"
                  Name="RatingCompanyROVO5">
    <AttrNames>
      <Item Value="QualId"/>
      <Item Value="Id"/>
      <Item Value="DisplayData"/>
      <Item Value="DisplayDatagroup"/>
      <Item Value="DisplayLabel"/>
      <Item Value="EngineDesc"/>
      <Item Value="Id"/>
      <Item Value="NodeId"/>
      <Item Value="ParentId"/>
      <Item Value="ContractStatus"/>
      <Item Value="IsAddEnginesAbove"/>
      <Item Value="IsAddEngines"/>
      <Item Value="IsAddDataGroups"/>
      <Item Value="DocDataType"/>
      <Item Value="IsDocEditable"/>
      <Item Value="DocStatus"/>
    </AttrNames>
    <Accessors>
      <Item Value="SubBaseEngines"/>
    </Accessors>
  </nodeDefinition>
  <nodeDefinition DefName="sni.irct.model.queries.readonly.BaseEngineROVO"
                  DiscrName="DisplayDatagroup" DiscrValue="1"
                  Name="RatingCompanyROVO55">
    <AttrNames>
      <Item Value="QualId"/>
      <Item Value="Id"/>
      <Item Value="DisplayData"/>
      <Item Value="DisplayDatagroup"/>
      <Item Value="DisplayLabel"/>
      <Item Value="EngineDesc"/>
      <Item Value="Id"/>
      <Item Value="NodeId"/>
      <Item Value="ParentId"/>
      <Item Value="ContractStatus"/>
      <Item Value="IsAddEnginesAbove"/>
      <Item Value="IsAddEngines"/>
      <Item Value="IsAddDataGroups"/>
      <Item Value="DocDataType"/>
      <Item Value="IsDocEditable"/>
      <Item Value="DocStatus"/>
```



```
    </AttrNames>
    <Accessors>
      <Item Value="DataGroups"/>
    </Accessors>
  </nodeDefinition>
  <nodeDefinition DefName="sni.irct.model.queries.readonly.DataGroupROVO"
                  Name="RatingCompanyROVO6">
    <AttrNames>
      <Item Value="Currency"/>
      <Item Value="CalcOprndGrpId"/>
      <Item Value="DisplayData"/>
      <Item Value="DisplayLabel"/>
      <Item Value="NodeId"/>
      <Item Value="ContractStatus"/>
      <Item Value="DocDataType"/>
      <Item Value="IsDocEditable"/>
      <Item Value="DocStatus"/>
    </AttrNames>
  </nodeDefinition>
```

If you need to add more info for the display attributes – please ***hand code*** these changes

3.11.5. Copy / Paste Elements of a Contract

Through the Contract Tree, it is possible to ***select*** and ***copy*** a section of a contract, and ***paste*** the selected section over to the same contract or another contract.

The following steps need to be taken for the copy – paste functionality:

1. ***Select a node*** (for copying) – a valid node in the tree is from qualifications or below
2. Press the ***Copy*** button
3. ***Select another node*** (for pasting the data)
4. Press the ***Paste*** [or ***Paste With Rates***] button

3.11.6. getSelection

A SelectionEvent is triggered when a node is selected in the contract tree as specified by the af:tree component:

```
<af:tree
      …

    selectionListener="#{pageFlowScope.ContractTreeHandler.getSelection}"
```

Since we are over-riding the default selection behavior of the adf tree we need to invoke another method to make our tree behave properly when we access it from the backing bean.

Technically, the default selectionListener for the tree should have been
"#{bindings.RatingCompanyROVO.treeModel.makeCurrent}"

The hack appears below.

```
/**
 * technically this method should be within our JSFUtils class of Foundation UI
 * the code has been copied from
 * http://andrejusb.blogspot.com/2009/11/tree-table-component-in-oracle-adf.html
 * sample application code
 * http://jdevsamples.googlecode.com/files/TreeComponents.zip
 * Also see POJO use case section from Frank Nimphius article available on ADF Code Corner -
 * How-to access the selected row data in a TreeTable or Tree
 * http://www.oracle.com/technetwork/developer-tools/adf/learnmore/26-get-selected-tree-node-data-169165.pdf
 */

public Object _invokeMethodExpression(String expr, Class returnType,
                                      Class[] argTypes, Object[] args)
{
  FacesContext fc = FacesContext.getCurrentInstance();
  ELContext elctx = fc.getELContext();
  ExpressionFactory elFactory =
    fc.getApplication().getExpressionFactory();
  MethodExpression methodExpr =
    elFactory.createMethodExpression(elctx, expr, returnType, argTypes);
  return methodExpr.invoke(elctx, args);
}
```

The *getSelection* method after making a call to the above method – `invokeMethodExpression` gets the instance of the RichTree component bound to the Contract Tree.

It first identifies which node in the Contract Tree has been selected.

Then depending on the applicable ROVO object used it populates the values for the following instance variable:

docDataTypeSelected	→ identifies the document data type of the selected node
nodeSelectedId	→ the id of the selected node [ID attribute for the applicable ROVORow]
nodeSelectedType	→ the type of node selected [contract node, qualification node, etc]
nodeSelectedTypeVal	→ an integer value for the type of node selected [Company – 0, Contract – 1, etc]
companyIdSelectedForCopyPaste	→ the id of the company

The value of ***docDataTypeSelected*** is retrieved by using the transient variable DocContentCode when the selection is made at the Document level; and DocDataType from the levels Qualification and below

The values used for ***nodeSelectedTypeVal*** come from the *IRCTRateCompanyConstants* as follows:

```
// node item types on the contract tree
public static final short COMPANY_ITEM_VAL                = 0;
public static final short CONTRACT_ITEM_VAL               = 1;
public static final short DOCUMENT_ITEM_VAL               = 2;
public static final short QUALIFICATION_ITEM_VAL          = 3;
public static final short ENGINE_ITEM_VAL                 = 4;
public static final short DATAGROUP_ITEM_VAL              = 5;
```


The flow of getSelection method is delineated below:

getSelection(SelectionEvent evt)

↓

`_invokeMethodExpression`

```
_invokeMethodExpression(
        "#{bindings.RatingCompanyROVO.treeModel.makeCurrent}",
        Object.class,
        new Class[]{ SelectionEvent.class },
        new Object[] { evt }
    );
```

↓

Get the RichTree component bound to the Contract Tree

```
RichTree aTree = this.getContractTree();
```

↓

Identify the collection of keys displayed in the Contract Tree

```
RowKeySet rowKeySet = aTree.getSelectedRowKeys();
```

↓

Iterate through the RowKeySet collection to identify which node has been selected

```
Iterator rksIterator = rowKeySet.iterator();
if (rksIterator.hasNext()) {
   List keys = (List) rksIterator.next();
   JUCtrlHierBinding treeBinding = null;
   treeBinding =
      (JUCtrlHierBinding)
                ((CollectionModel)aTree.getValue()).getWrappedData();
   JUCtrlHierNodeBinding nodeBinding =
        treeBinding.findNodeByKeyPath(keys);

   String nodeStuctureDefname =
        nodeBinding.getHierTypeBinding().getStructureDefName();
```

↓

Based on its name populate the instance variables:

docDataTypeSelected
nodeSelectedId
nodeSelectedType
nodeSelectedTypeVal
companyIdSelectedForCopyPaste

```
   if(nodeStuctureDefname.equals(IRCTRateCompanyConstants.DOCUMENT_VIEW)) {
    . . .
   } else
 if
 (nodeStuctureDefname.equals(IRCTRateCompanyConstants.QUALIFICATION_VIEW)) {
     . . .
   }
  . . .
```


3.11.7. Copying a Node

After the user has selected the node to copy, clicking on the '**Copy**' button will activate the ***copyNode*** method in the ContractTreeHandler class.

This method primarily stores the information as available for copying for valid nodes – those which exist below the Document level – and then sets the corresponding Paste buttons to be visible and enabled

The following instance variables are used to store the information for copying:

- nodeSelectedIdForCopy → id of the node selected after copy button is pressed
- nodeSelectedForCopyType → type of node selected after copy is pressed
- nodeSelectedTypeValForCopy → for RCTContractItem -- type; copy
- docDataTypeSelectedForCopy → doc data type selected for copy

Note that depending on the data type of the Document associated with the selected node, the text for the paste buttons differ once a valid node is *copied*.

Associated Document Data Type	***"Paste"* button text**	***"Paste with"* button text**
TNST	Paste Without Transits	Paste With Transits
RTNG	Paste Without Data	Paste With Data
LKUP SRV PRD	Paste Without Rates	Paste With Rates

3.11.8. Pasting the Contents

Having selected the contents to be copied by pressing on the *Copy* button, and after selecting another node to which the contents need be copied under – the user can then click on the *Paste* buttons.
The click on the '**Paste..**' button activates the ***pasteSelectedNode*** method in the ContractTreeHandler class.

This method sets the instance variables used for storing the pasting information based on the last SelectionEvent of the selected node [to which the items need to be pasted under]:

nodeSelectedIdForPaste → id of the node selected after a paste button is pressed
nodeSelectedForPasteType → type of node selected after a paste button is pressed
nodeSelectedTypeValForPaste → for RCTContractItem -- type; paste

The method then generates two RCTContractItem objects – one for the copy, one for the paste.

It then calls upon the ***pasteItemUnder*** method in the IRCTRateCompanyService – which passes along the information to make a CORBA call for pasting the information.


```
RCTContractItem copiedItem = new RCTContractItemImpl();
copiedItem.setItemID(new Long(nodeSelectedIdForCopy).longValue());
copiedItem.setItemType(nodeSelectedTypeValForCopy);


RCTContractItem pasteUnderItem = new RCTContractItemImpl();
pasteUnderItem.setItemID(new Long(nodeSelectedIdForPaste).longValue());
pasteUnderItem.setItemType(nodeSelectedTypeValForPaste);


// make the CORBA calls
OperationBinding operationBinding =
  ADFBindingUtils.getBindingContainer().getOperationBinding("pasteItemUnder");

operationBinding.getParamsMap().put("pCopiedItem", copiedItem);
operationBinding.getParamsMap().put("pPasteUnderItem", pasteUnderItem);
operationBinding.getParamsMap().put("pIncludeRates", withSelected);
operationBinding.getParamsMap().put("pCompanyId", companyIdSelectedForCopyPaste);

operationBinding.execute();
```

Within the IRCTRateCompanyServiceImpl class the ***pasteItemUnder*** method uses the ***RatingAdminManager*** class to make the CORBA call to paste items in the Contract Tree – which are essentially insert operations in the database.

```
public void pasteItemUnder(RCTContractItem pCopiedItem,
                           RCTContractItem pPasteUnderItem,
                           boolean pIncludeRates, String pCompanyId) {

        . . .
        RatingAdminManager ratingAdminManager = new RatingAdminManager();

        . . .

        ratingAdminManager.pasteItemUnder(pCopiedItem, pPasteUnderItem,
                                          pIncludeRates, usrInfo);
        . . .
}
```

Two Exceptions may be thrown during this ‘pasting’ process:

- IRCTSystemException
- IRCTValidationException

In the event that an exception is indeed thrown an appropriate error code and error message is sent back to the front end.

```
. . .
String errorMessage = IRCTPropertyUtils.getIRCTMessagesPropertyValue(e.getMessage());

errorMessage = (errorMessage == null)?
                    "Unknown error while copying - error id " + e.getMessage():
                    errorMessage;

throw new JboException(errorMessage);

. . .
```

e.getMessage() is the error code returned by the failed CORBA operation
The String ***errorMessage*** is error message sent back to the end user.

The text of these messages – and their codes – is stored in the ***IRCTMessages.properties*** file located in the *IRCTResources project*.

The codes have been taken from *ircterrors.h* from the legacy application. The following error codes and related text strings applicable for the Contract Tree Copy / Paste functionality appear below:

- 26066=The copied item is invalid. Only Qualifications, Engines, and Data Groups may be copied.
- 26070=The item to paste under is invalid. You may only paste under Documents, Qualifications, and Engines.
- 25718=The engine type is invalid. Only engine types that are valid parents of the engine can be added above.
- 25722=The engine type is invalid. Only engine types that are Collective may be added above another engine.
- 25726=The engine type is invalid. Engine types that are Dependent may not add an engine above them.
- 26074=A copied Qualification may only be pasted under a Document.
- 26078=The copied Engine can NOT be pasted under the Qualification.
- 26082=The copied Engine can NOT be pasted under the Engine.
- 26086=A copied Data Group may only be pasted under a Base Engine.
- 26090=A copied Data Group may only be pasted under: 1. a Base Engine of the same type as the parent Base Engine. 2. a Base Engine that has the same Index Types for Origin, Destination, and Stops as the parent Base Engine.
- 26094=Unable to create, the document is not in a creatable state.
- 26098=The copied item is a duplicate to that of another item in the tree. Change the precedence and try again.
- 26102=A copied Engine may only be pasted under a Qualification or another Engine.
- 26386=The copied qualification is a duplicate to another qualification in the document. Please verify that the precedence and description are not used within the document and try again.
- 26858=A copied item may only be pasted under a document of the same data type.
- #The Document of the Copied Item must be the same type as the Document of the paste under item
- 27562=The copied item is invalid. You cannot paste an item with the status of Deleted.
- 30646=A copied Data Group cannot be pasted under a Base Engine in a Routing Guide Document.
- 2119=Unable to communicate with the Server. Please contact your System Administrator.
- 1674=Cannot perform the desired action because the contract is inactive
- 1853=Unable to create, the document is not in a creatable state.
- 1852=Unable to update, the document is not in an updatable state.
- 1854=Unable to delete, the document is not in a deletable state.

Since it is quite likely that the collection of error codes for the Copy / Paste functionality is not complete – in the event that *IRCTPropertyUtils.getIRCTMessagesPropertyValue(e.getMessage());* returns a ***null*** – an 'Unknown Error..'' along with the error code is sent back to the user,

It is expected that in the event that such an *unknown error* occurs for the paste operation – the user will contact support with the error id so that the appropriate information may be updated in the IRCTMessages.properties file.

When the pasting operation completes – the paste buttons are disabled and made invisible again.

3.11.9. DisplayLabel & DisplayData attributes

The following table describes which attributes are used for the DisplayLabel and DisplayData output text within the ***nodeStamp*** facet in the Contract Tree:

Level [ROVO]	DisplayLabel	DisplayData
Company [RatingCompanyROVO]	Name	• Id • ExpDays • Fc → if not null • Phone → if not null
Contract [ContractROVO]	Name	• Id • ScdNameData • HistValue • TypeValue • StatusValue • displays *AUDIT* if AUD_FLG = 1
Document [DocumentROVO]	DocContentCodeValue EffDt StatusValue	• Id • DocTypeValue • DistPackValue • DistVers • DistCalcType • UsageValue • ExpDate • StatusValue • CurrencyFrequencyValue • ExclCanBorderMiles • ExclMilesWithinMex
Qualifications [QualificationROVO]	QualDesc	• Id • Prec • StatusValue • RevValue
Collective Engines [CollectiveEngineROVO]	EngineDesc	• Id • Prec • DocContentCode • EngineCodeValue • StatusValue • RevValue → displays if *REV* is not empty • displays *AUDIT* if AUD_FLG = 1
Base Engines [BaseEngineROVO]	EngineDesc	• Id • Prec • DocContentCode • EngineCodeValue • StatusValue • RevValue • Displays *CON REF* if RequireContractRef is 1 • Displays *BandCodeValue concatenated with*

		UomDesc if RequireBandCode is 1 • Displays *TierValue* if RequireTier is 1 • Displays *"PRIORITY | DURATION"* if IncludePriorityDuration is 1 • Displays *PRICING IDS* if IncludePricingBid is 1 • Displays *TranistTypeInfo* if RequireTransit is 1 • Displays OrigValue if not null • Displays DestValue if not null
Data Groups [DataGroupsROVO]	Description Currency	• Records • StatusValue • RevValue

3.11.10. Dynamic Command Links

The dynamic command links displayed at each level in the node direct the user to leading screens to perform further operations.

The links are displayed based on the users Security Roles, the type of Contracts (Active or Inactive), the data types of the Documents, etc – as delineated in the following.

Contract Tree Dynamic Command Links		
Text	*Rendered If*	
	Conditions	*Security Role*
Company		
add default transit contract	Name of Company is found -- ALWAYS RENDERED	RoutingGuideAnalyst, RateAnalyst, TransitAnalyst
add contract	Name of Company is found -- ALWAYS RENDERED	RoutingGuideAnalyst, RateAnalyst, TransitAnalyst
Contract		
add document	Contract Hist Is not null Contract Status is 'ACTV'	RoutingGuideAnalyst, RateAnalyst, TransitAnalyst
edit / view	Contract Hist Is not null Contract Status is 'ACTV' Contract Name is **NOT** 'DEFAULT TRANSIT'	RoutingGuideAnalyst, RateAnalyst, TransitAnalyst
inactivate	Contract Hist Is not null Contract Status is 'ACTV' Contract Name is **NOT** 'DEFAULT TRANSIT'	RoutingGuideAnalyst, RateAnalyst, TransitAnalyst
delete	Contract Hist Is not null Contract Status is 'ACTV'	RoutingGuideAnalyst, RateAnalyst, TransitAnalyst
view	Contract Hist Is not null	
	Contract Name **IS** *'DEFAULT TRANSIT'*	RoutingGuideAnalyst, RateAnalyst, TransitAnalyst
		RoutingGuideAnalystOnly, RateAnalystOnly, TransitAnalystOnly

Contract Tree Dynamic Command Links		
Text	*Rendered If*	
	Conditions	*Security Role*
Document		
add qualification	Document Type is not null Contract Status is 'ACTV' REQ_CRE_FLG of corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
edit / view	Document Type is not null Contract Status is 'ACTV' UPD_FLG or UPD_EFF_DT_FLG or UPD_EXPR_DT_FLG of corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
audit / view	Document Type is not null Contract Status is 'ACTV'	
	UPD_FLG or UPD_EFF_DT_FLG or UPD_EXPR_DT_FLG of corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalystReadOnly, Document Data Type is 'TNST' and TransitAnalystReadOnly, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalystReadOnly
	UPD_FLG or UPD_EFF_DT_FLG or UPD_EXPR_DT_FLG of corresponding DOC_STA is 1	
promote	Document Type is not null Contract Status is 'ACTV' PROM_FLG of corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
delete	Document Type is not null Contract Status is 'ACTV' DEL_FLG of corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
view	Document Type is not null Contract Status is 'INAC'	RoutingGuideAnalystOnly, RateAnalystOnly, TransitAnalystOnly

Contract Tree Dynamic Command Links		
Text	*Rendered If*	
	Conditions	*Security Role*
Qualification		
add engine	Qualification Description is not null Contract Status is 'ACTV' UPD_FLG of the Document's corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
audit / view	Qualification Description is not null Contract Status is 'ACTV' UPD_FLG of the Document's corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalystReadOnly, Document Data Type is 'TNST' and TransitAnalystReadOnly, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalystReadOnly
	UPD_FLG of the Document's corresponding DOC_STA is 0	
edit / view	Qualification Description is not null Contract Status is 'ACTV' UPD_FLG of the Document's corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
delete	Qualification Description is not null Contract Status is 'ACTV' UPD_FLG of the Document's corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
view	Qualification Description is not null Contract Status is INAC'	Document Data Type is 'RTNG' and RoutingGuideAnalystReadOnly, Document Data Type is 'TNST' and TransitAnalystReadOnly, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalystReadOnly
Collective Engine		
add engine	Engine Description is not null Engine's Parent Id [PRNT_CALC_OPR_ID] IS NULL Contract Status is 'ACTV' UPD_FLG of the Document's corresponding DOC_STA is 1 Corresponding Engine Type has CRE_BAS_FLG set to 1	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
add engine above	Engine Description is not null Engine's Parent Id [PRNT_CALC_OPR_ID] IS NULL UPD_FLG of the Document's corresponding DOC_STA is 1 Contract Status is 'ACTV'	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
edit / view	Engine Description is not null Engine's Parent Id [PRNT_CALC_OPR_ID] IS NULL Contract Status is 'ACTV' UPD_FLG of the Document's corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
audit / view	Engine Description is not null Engine's Parent Id [PRNT_CALC_OPR_ID] IS NULL Contract Status is 'ACTV' UPD_FLG of the Document's corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalystReadOnly, Document Data Type is 'TNST' and TransitAnalystReadOnly, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalystReadOnly
	UPD_FLG of the Document's corresponding DOC_STA is 0	
delete	Engine Description is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Engine's Parent Id [PRNT_CALC_OPR_ID] IS NULL Contract Status is 'ACTV'	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
view	Engine Description is not null Engine's Parent Id [PRNT_CALC_OPR_ID] IS NULL Contract Status is 'INAC'	Document Data Type is 'RTNG' and RoutingGuideAnalystReadOnly, Document Data Type is 'TNST' and TransitAnalystReadOnly, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalystReadOnly

Contract Tree Dynamic Command Links		
Text	*Rendered If*	
	Conditions	*Security Role*
Base Engine		
add engine	Engine Description is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Engine's Parent Id [PRNT_CALC_OPR_ID] IS NOT NULL Contract Status is 'ACTV' Corresponding Engine Type has CRE_BAS_FLG set to 1	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
add engine above	Engine Description is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Engine's Parent Id [PRNT_CALC_OPR_ID] IS NOT NULL Contract Status is 'ACTV'	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
add data group	Engine Description is not null Engine's Parent Id [PRNT_CALC_OPR_ID] IS NOT NULL UPD_FLG of the Document's corresponding DOC_STA is 1 Contract Status is 'ACTV' Corresponding Engine Type has CRE_OPRND_FLG set to 1 Corresponding Document Data Type IS NOT 'RTNG'	Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
edit / view	Engine Description is not null Engine's Parent Id [PRNT_CALC_OPR_ID] IS NOT NULL Contract Status is 'ACTV' UPD_FLG of the Document's corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
audit / view	Engine Description is not null Engine's Parent Id [PRNT_CALC_OPR_ID] IS NOT NULL Contract Status is 'ACTV'	
	UPD_FLG of the Document's corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalystReadOnly, Document Data Type is 'TNST' and TransitAnalystReadOnly, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalystReadOnly
	UPD_FLG of the Document's corresponding DOC_STA is 0	
delete	Engine Description is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Engine's Parent Id [PRNT_CALC_OPR_ID] IS NOT NULL Contract Status is 'ACTV'	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
view	Engine Description is not null Engine's Parent Id [PRNT_CALC_OPR_ID] IS NOT NULL Contract Status is 'INAC'	Document Data Type is 'RTNG' and RoutingGuideAnalystReadOnly, Document Data Type is 'TNST' and TransitAnalystReadOnly, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalystReadOnly

Contract Tree Dynamic Command Links		
Text	*Rendered If*	
	Conditions	*Security Role*
Data Groups		
undo rates	DataGroups Currency is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Contract Status is 'ACTV'	Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
import rates	DataGroups Currency is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Contract Status is 'ACTV'	Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
export rates	DataGroups Currency is not null Contract Status is 'ACTV'	Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
add rates	DataGroups Currency is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Contract Status is 'ACTV'	Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
edit / view rates	DataGroups Currency is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Contract Status is 'ACTV'	Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
audit / view rates	DataGroups Currency is not null Contract Status is 'ACTV'	
	UPD_FLG of the Document's corresponding DOC_STA is 1	Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
	UPD_FLG of the Document's corresponding DOC_STA is 0	Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalystReadOnly
import transits	DataGroups Currency is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Contract Status is 'ACTV'	Document Data Type is 'TNST' and TransitAnalyst
export transits	DataGroups Currency is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Contract Status is 'ACTV'	Document Data Type is 'TNST' and TransitAnalyst
add transits	DataGroups Currency is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Contract Status is 'ACTV'	Document Data Type is 'TNST' and TransitAnalyst
edit / view transits	DataGroups Currency is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Contract Status is 'ACTV'	Document Data Type is 'TNST' and TransitAnalyst
audit / view transits	DataGroups Currency is not null Contract Status is 'ACTV'	
	UPD_FLG of the Document's corresponding DOC_STA is 1	Document Data Type is 'TNST' and TransitAnalyst
	UPD_FLG of the Document's corresponding DOC_STA is 0	Document Data Type is 'TNST' and TransitAnalystReadOnly

Contract Tree Dynamic Command Links		
Text	*Rendered If*	
	Conditions	*Security Role*
import data	DataGroups Currency is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Contract Status is 'ACTV'	Document Data Type is 'RTNG' and RoutingGuideAnalyst
export data	DataGroups Currency is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Contract Status is 'ACTV'	Document Data Type is 'RTNG' and RoutingGuideAnalyst
add data	DataGroups Currency is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Contract Status is 'ACTV'	Document Data Type is 'RTNG' and RoutingGuideAnalyst
edit / view data	DataGroups Currency is not null UPD_FLG of the Document's corresponding DOC_STA is 1 Contract Status is 'ACTV'	Document Data Type is 'RTNG' and RoutingGuideAnalyst
audit / view data	DataGroups Currency is not null Contract Status is 'ACTV'	
	UPD_FLG of the Document's corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalyst
	UPD_FLG of the Document's corresponding DOC_STA is 0	Document Data Type is 'RTNG' and RoutingGuideAnalystReadOnly
edit / view	DataGroups Currency is not null Contract Status is 'ACTV' UPD_FLG of the Document's corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
view rates	DataGroups Currency is not null Contract Status is 'INAC'	Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
view data	DataGroups Currency is not null Contract Status is 'INAC'	Document Data Type is 'RTNG' and RoutingGuideAnalyst
view transits	DataGroups Currency is not null Contract Status is 'INAC'	Document Data Type is 'TNST' and TransitAnalyst
delete	DataGroups Currency is not null Contract Status is 'ACTV' UPD_FLG of the Document's corresponding DOC_STA is 1	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
view	DataGroups Currency is not null	
	Contract Status is 'INAC'	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst
	UPD_FLG of the Document's corresponding DOC_STA is 0	Document Data Type is 'RTNG' and RoutingGuideAnalyst, Document Data Type is 'TNST' and TransitAnalyst, Document Data Type is 'LKUP' or 'PRD' or 'SRV and RateAnalyst

3.11.11. Pass the Map Please

Most of the Dynamic Command Links possess an ***actionListener*** attribute which calls upon the ***contractTreeMap*** method in the ContractTreeHandler class. This method's sole purpose is to identify information in the tree upto the selected level in the node where the link appears – generate a HashMap for that data – and then pass on that HashMap in pageFlowScope to the next leading screen or taskflow.

In the event the HashMap is being passed to a taskflow – then that task flow has to define the HashMap as one of its input parameters.

The process of identifying the information for generating the HashMap initially is very much the same as in the *getSelection* method in that we get the *RowKeySet* – iterate through it and check for the *nodeStuctureDefname* value.

The Map is put into pageFlowScope using the following:

```
AdfFacesContext.getCurrentInstance().getPageFlowScope().put("contractTreeDataMap", theMap2Pass);
```

[Potential] Keys {and their values} in the ContractTreeMap include the following:

IRCTRateCompanyConstants.CONTRACT_TREE_PATH --> value is the Tree Path

IRCTRateCompanyConstants.CPY_ID --> value is the company id for rating company at the root node
IRCTRateCompanyConstants.BUS_ID --> value is the business id for rating company at the root node

IRCTRateCompanyConstants.CTRC_ID --> value is the contract id
IRCTRateCompanyConstants.CTRC_NAME --> value is the name of the contract [ctrc.ctrc_desc]
IRCTRateCompanyConstants.CTRC_LABEL --> value is the label displayed at the node for the Contract

IRCTRateCompanyConstants.DOC_ID --> value is the document id
IRCTRateCompanyConstants.DOC_STATUS --> value is the document status [doc_sta.DOC_STA_CD]
IRCTRateCompanyConstants.DOC_TYPE --> value is the document type [doc_typ.DOC_CD]
IRCTRateCompanyConstants.DOC_DESC --> value is the document description [doc_typ.DOC_CTNT_CD]
IRCTRateCompanyConstants.DOC_EFF_DATE --> value is the effective date for the document
IRCTRateCompanyConstants.DOC_LABEL --> value is the label displayed at the node for the Document

IRCTRateCompanyConstants.QUAL_ID --> value is the Qualification id
IRCTRateCompanyConstants.QUAL_DESC --> value is the qualification description [qual_desc]
IRCTRateCompanyConstants.QUAL_LABEL --> value is the label displayed at the node for the Qualification

IRCTRateCompanyConstants.CALLER_ENGINE_ID --> value is top level engine id - the caller engine id [top most collective engine]
IRCTRateCompanyConstants.ENGINE_CODE --> value is the type of for the applicable node [calc_opr_typ.CALC_OPR_CD -- DZBM, BVBR, DBND, DZBV, TREF, BAND...]
IRCTRateCompanyConstants.ENGINE_CLASS_CODE --> value is the Engine class type for the applicable node [calc_opr_typ.CALC_OPR_CLA_CD -- COLL, BASE, DPND]
IRCTRateCompanyConstants.COLL_ENGINE_DESC --> value is the description [i.e calc_opr.CALC_OPR_DESC] for the Collective Engine
IRCTRateCompanyConstants.COLL_ENG_LABEL --> value is the label displayed at the node for the CollectiveEngine

IRCTRateCompanyConstants.BASE_ENG_ID --> value is the id of the LOWEST BaseEngine within the tree
IRCTRateCompanyConstants.BASE_ENGINE_DESC --> value is the description [i.e calc_opr.CALC_OPR_DESC] for the LOWEST BaseEngine within the tree
IRCTRateCompanyConstants.BASE_ENG_LABEL --> value is the label displayed at the LOWEST BaseEngine within the tree

IRCTRateCompanyConstants.DATA_GROUP_ID --> value is the id of the DataGroup
IRCTRateCompanyConstants.DATA_GROUP_DESC --> value is the description [i.e uom_typ.UOM_DESC] for the DataGroup
IRCTRateCompanyConstants.DATA_GROUP_LABEL --> value is the label displayed at the node for DataGroup
IRCTRateCompanyConstants.CONTRACT_TREE_LINK_INFO --> value is the text of the link clicked
IRCTRateCompanyConstants.IS_READ_ONLY --> value is either "true" or "false" depending on if link contains the text "add" or "edit" ("true") or not ("false")

3.12. Data Volume

Use Case ID	Priority	Usage	Time	Throughput

3.13. Usage Description and Quantification

	Projected High	Projected Low
Current Load Counts in Production:		
Expected Concurrent Users:		
Max Concurrency Levels at Peak Times:		
Peak Access Times:	7am-5pm	
Connection Speeds used:		

3.14. Connectivity and Bandwidth

N/A

3.15. Security

4. Milestones (risks mitigation, tracking, etc)

Milestone	Date	Owner

5. Revision History

Version #	Description	Author	Date
1.0	Baseline		
2.0	Added Milestones		

