d

TOne Routing Technical Document

Contents

[Document Purpose 2](#_Toc418615185)

[Routing Overview 3](#_Toc418615186)

[Module Dependencies 3](#_Toc418615187)

[Module Components 3](#_Toc418615188)

[Visual Studio Projects 3](#_Toc418615189)

[Full Routing Business Process 5](#_Toc418615190)

[Build LCR and Routes Sub Process 7](#_Toc418615191)

[Build Routes Activity 7](#_Toc418615192)

[Build Routes Activity General Execution 7](#_Toc418615193)

[Build Routes Activity Entities 9](#_Toc418615194)

[Differential Routing Business Process 13](#_Toc418615195)

# Document Purpose

The purpose of this document is to discuss the new implementation of the Route Module. This includes:

* LCR
* Route Build
* Route Sync

# Module Overview

The Route module is responsible for building the most effective routes that should be used and synchronize it with the Switch(es).

The building of the routes is basically dependent on two main things:

* LCR (Least Cost Route): this is the output of the Effective Code, Zone, and Rate. Either these entities belong to the system (sales part) or to the suppliers
* Manual Routing: this is represented by Tone’ users defining Routing Rules:
  + Route Block
  + Route Override
  + Priority
  + …

## Module Components

The following table shows the different components of the Routing Module:

|  |  |
| --- | --- |
| Component Name | Type |
| Full Routing Business Process | Workflow |
| Differential Routing Business Process | Workflow |
| Route Rule Management | User Interface |
| Route Manager | User Interface |
| Rate Planning & Sale LCR | User Interface |

## Visual Studio Projects

The following table shows the visual studio projects of the Routing Module:

|  |  |
| --- | --- |
| VS Project Name | Notes |
| TOne.LCR.Entities |  |
| TOne.LCR.Data |  |
| TOne.LCR.Data.SQL |  |
| TOne.LCR.Business |  |
| TOne.LCRProcess | Routing Workflows |
| TOne.LCRProcess.Activities | Routing Workflow Activities |
| TOne.LCRProcess.Arguments | Routing Workflow In/Out Arguments |
| TOne.LCR.Web | Routing UI and WebAPI |

These projects are located in TOneV2 solution under “C:\TFS\TOneV2\Code\TOneV2”

## Module Dependencies

The following table shows the dependencies of the Routing Module:

|  |  |
| --- | --- |
| Dependency Name | Location |
| TOne Main Module & Business Entity | TOneV2 solution |
| Vanrise.Common & Vanrise.Caching | Vanrise Framework solution |
| Vanrise.Data & Vanrise.Data.SQL | Vanrise Framework solution |
| Vanrise.Queueing | Vanrise.BusinessProcess Framework solution |
| Vanrise.BusinessProcess, Vanrise.BusinessProcess.Entities, & Vanrise.BusinessProcess.WFActivities | Vanrise.BusinessProcess Framework solution |

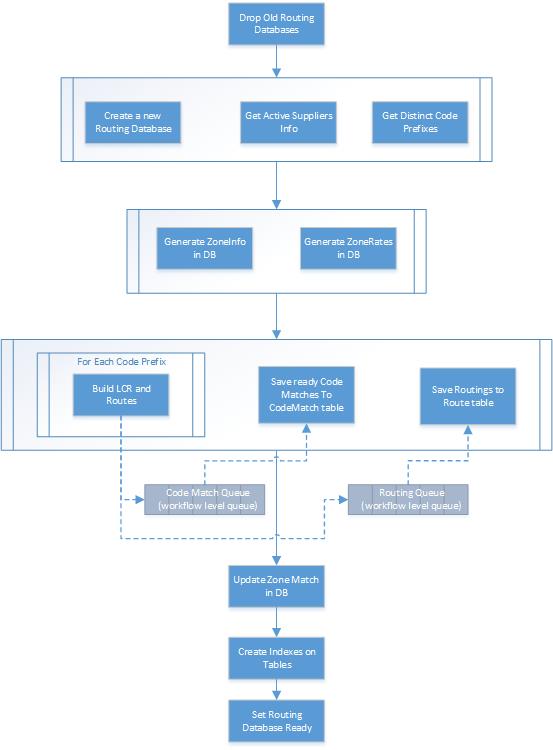
## Required Databases

|  |
| --- |
| Database Connection String Name |
| MainDBConnString |
| TransactionDBConnString |
| LogDBConnString |
| RoutingDBConnStringTemplateKey |

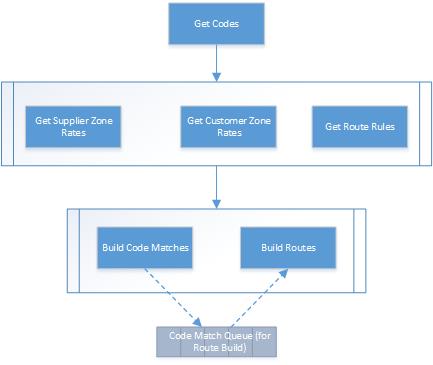
# Full Routing Business Process

The Full Routing Business Process is represented in following table:

|  |  |
| --- | --- |
| Billing CDRs Generation Process | |
| Job Summary | Create a full Routing database based on a specific Code/Zone/Rate Effective time or Future Routing |
| Trigger | Time Schedule/User Trigger |
| Input | * EffectiveTime: the time at which the source Code/Zone/Rate should be effective. And the Routing Database would be effective at this time * IsFuture: a flag indicating to use Code/Zone/Rate that will be effective in the future instead of relying on the EffectiveTime argument * IsLcrOnly: Build only LCR (i.e. Supply Rates) without the Routes |
| Output | Create and fill a new Routing database |
| Parallel Restriction |  |
| Tables Affected | All tables in the new Routing database: Route, CodeMatch, ZoneMatch , CustomerZoneRate, SupplierZoneRate, ZoneInfo |
| Sequence of Execution | 1. Drop Old Routing DBs (based on configuration) and Create a new Routing database 2. Get Active Suppliers and Generate ZoneInfo, CustomerZoneRate, and SupplierZoneRate tables 3. Get the list of Code Prefixes from the Code table based on Code Prefix length configuration. The purpose of the code prefixes is to divide the codes into groups of codes each of which allows independent processing and Code Match retrieval 4. Process each Group of Codes based on the Code Prefix and produce the list of Code Match and the list of Route 5. Other activities store the produced Code Matches and Routes in parallel 6. Generate Zone Match table, create indexes on the tables and set the database as ready   Check below diagram |



## Build LCR and Routes Sub Process



This sub process:

1. takes Routing Database Id, Code Prefix as input
2. retrieves the List of Codes and Zone Ids based on the Code Prefix from the Code table
3. retrieves the list of Supplier Zone Rates and Customer Zone Rates based on the Zone Ids from the Routing Database
4. retrieves the list of Route Rules (e.g. Override Route, Block Route…) based on the Code Prefix and Zone Ids from Route Rule tables
5. Builds Code Matches and Routes and send them to the workflow queues to store them in the database (as shown in the main workflow diagram)

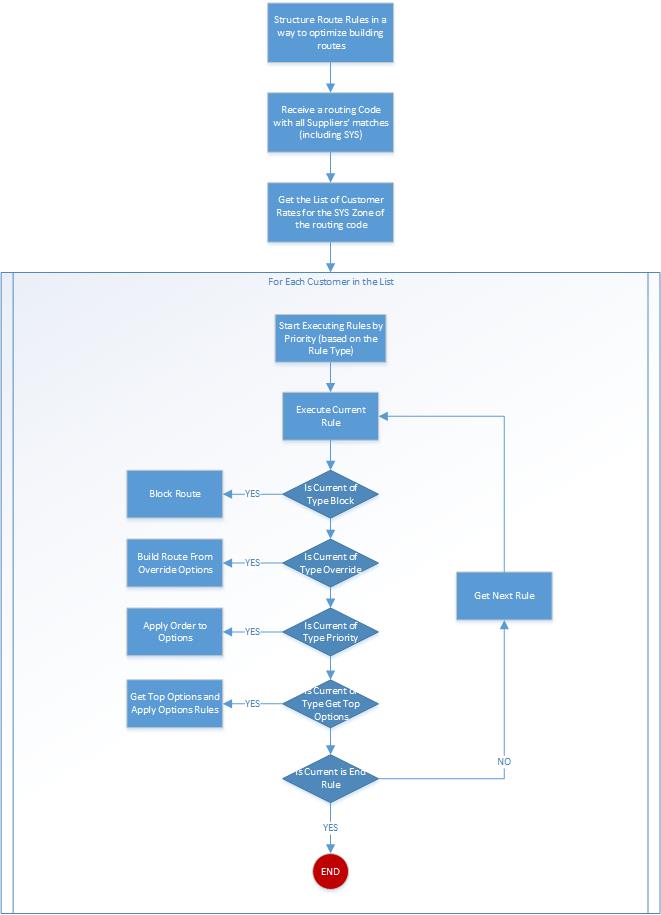
## Build Routes Activity

The Build Routes activity is the most complicated activity in the Routing process. The main logic is encapsulated in this activity. This activity:

* Takes Route Rules, Supplier Zone Rates, and Customer Zone Rates as input
* Has an Input Queue of Code Match produced by the Build Code Match activity
* Produce the Routes and put them in the Route Output Queue

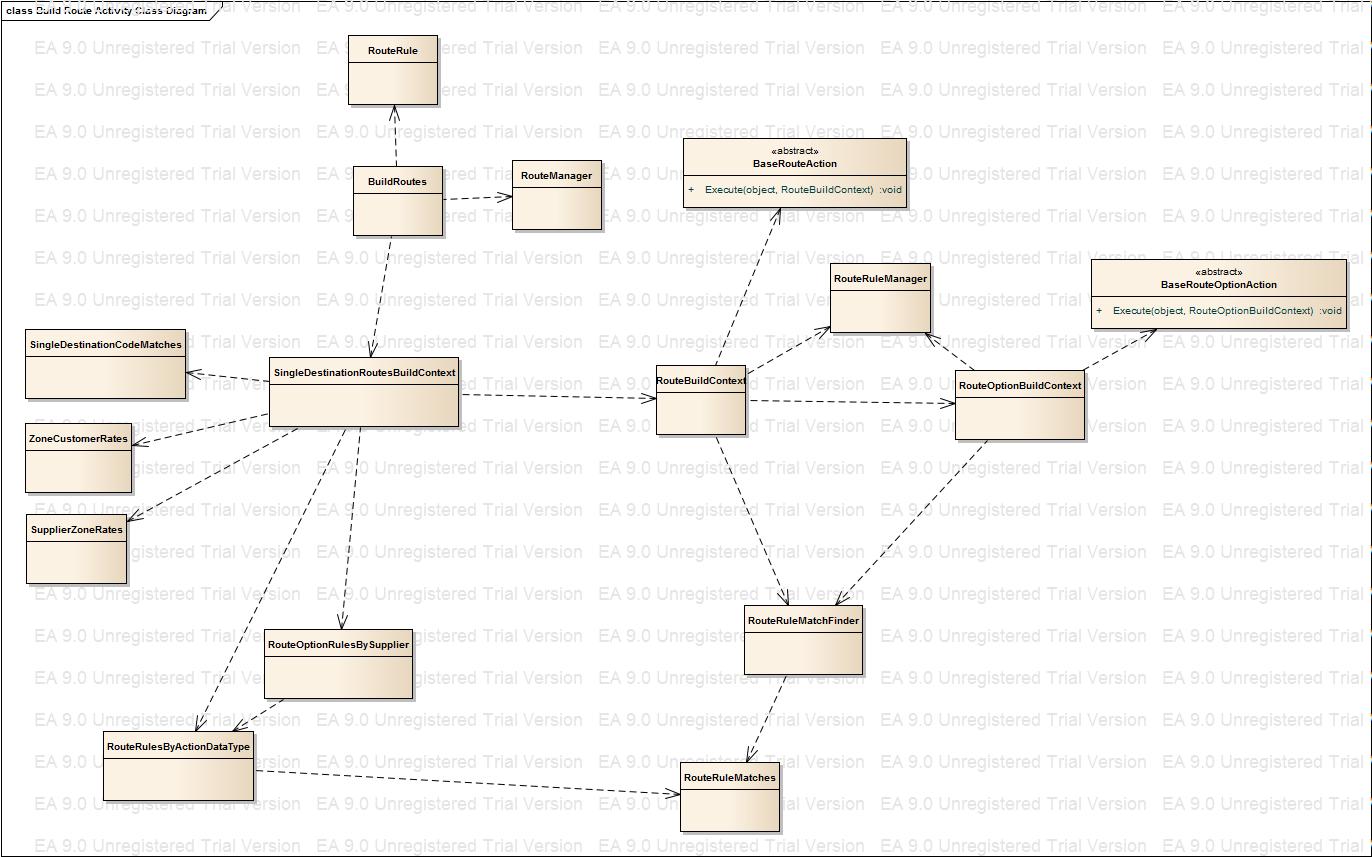
### Build Routes Activity General Execution

The following diagram shows the general execution logic of this activity:

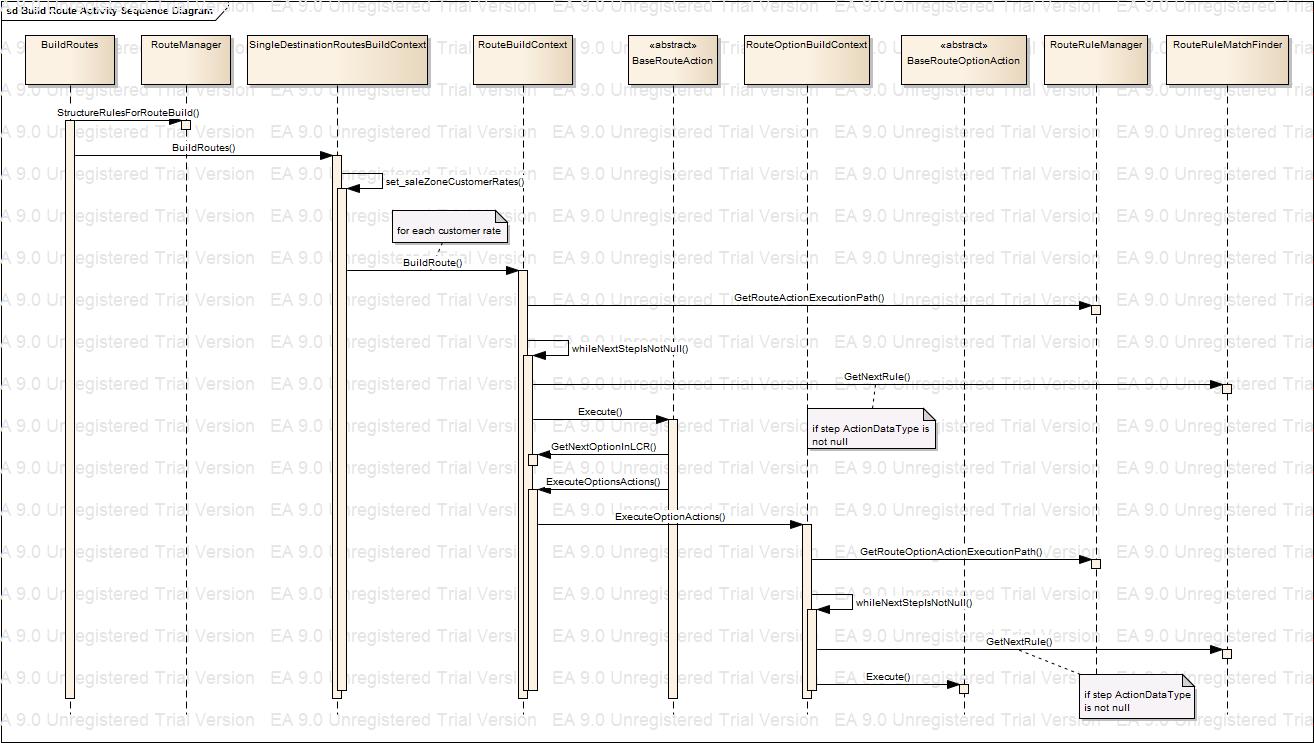


### Build Routes Activity Entities

The Build Routes activity relies on the entities shown in the following class diagram:



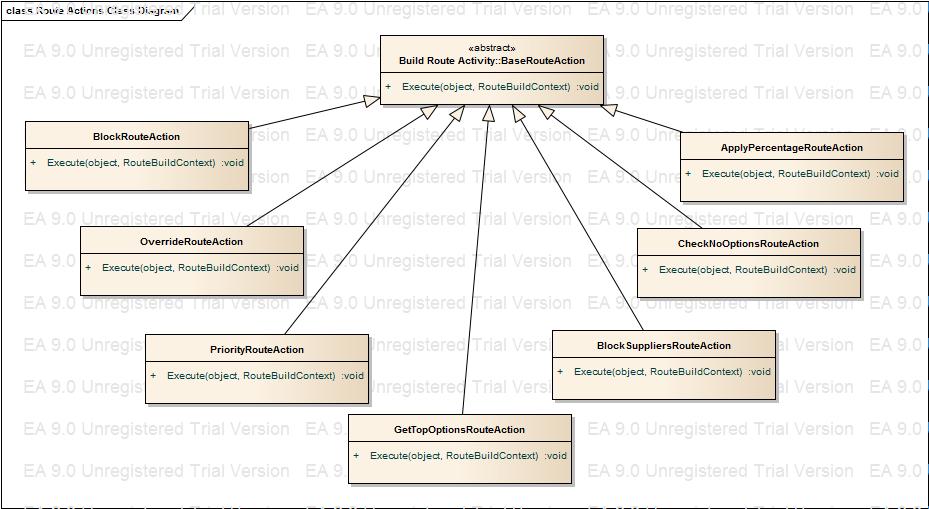
The following diagram shows the main sequence interaction between them:

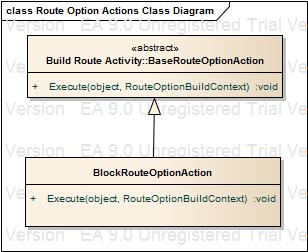


The following table gives more information about these entities:

|  |  |  |
| --- | --- | --- |
| Entity Name | Main Method | Description |
| RouteRule |  | The definition of the Route Rule. Each object consists of:   * CarrierAccountSet: Customer(s), Supplier(s)… * CodeSet: Code(s), Zone(s) * Type: RouteRule, RouteOptionRule * ActionData: Override Route, Block Route… |
| BaseRouteAction | Execute | Represents a Route-level action behavior (e.g. Override Route, Block Route…) |
| BaseRouteOptionAction | Execute | Represents a Route Option-level action behavior (e.g. Block Supplier) |
| RouteManager | StructureRulesForRouteBuild | * Get list of Route Rules as input * Structure the rules in two list: Route Rules, and Route Option Rules. The purpose of this structuring is to optimize lookup * The Route Rules list is structured based on Action Type in a dictionary * The Route Option Rules is structured based on Action Type and then on Supplier Id |
| SingleDestinationRoutesBuildContext | BuildRoutes | This entity is responsible for building the routes for a single route code:   * It takes the route code and all supplier code matches as input (encapsulated in SingleDestinationCodeMatches) * It takes zone rates and structured route and route option rules as input * It retrieves the list of customers that have active rates on the corresponding zone rates * It creates an instance of RouteBuildContext for each customer and build the route |
| RouteBuildContext | BuildRoute | This entity is responsible for building one single route (for a single route code and a single customer):   * It iterates over the route rules based on their action types by priority * It executes the route rule if it should be (either the action data is not required or a match action data is available) * The execution of each route rule is defined according to its type:   + Block Route: remove all options and set the route as blocked   + Override Route: validate and define route options from the Override Rule options   + Priority Rule: change the order of the options   + Get Top Options: restrict the number of options, and apply Route Options Rules if any   + … * If the route rule is executed, it checks if it is an end rule. If so, it stops otherwise it moves to the next route rule by priority |
| RouteOptionBuildContext | ExecuteOptionActions | This entity is responsible of applying rules defined on the Route Option level (e.g. Supplier Block). It can called by any BaseRouteAction implementation through the RouteBuildContext. ExecuteOptionsActions. |
| RouteRuleMatches |  | This entity is used to structure the Route Rules based on their definition. It is used in conjunction with the RouteRuleMatchFinder entity to make Rule execution optimized. It divides the Route Rules across the following lists:   * RulesByMatchCodes * RulesByMatchCodeAndSubCodes * RulesByMatchZones * RulesMatchingAllZones |
| RouteRuleMatchFinder | GetNext | This entity is used on top of the RouteRuleMatches entity. It is used to retrieve the match Route Rules for a specific Code |

The class diagrams of BaseRouteAction and BaseRouteOptionAction implementations are respectively shown below:





# Differential Routing Business Process