# BusBuddy

Generated by Doxygen 1.8.3.1

Fri Apr 26 2013 21:31:26

# Contents


**1 Namespace Index** **1**

1.1 Packages . . . . . 1

**2 Hierarchical Index** **3**

2.1 Class Hierarchy . . . . . 3

**3 Class Index** **7**

3.1 Class List . . . . . 7

**4 Namespace Documentation** **13**

4.1 Package alert.client . . . . . 13

4.1.1 Detailed Description . . . . . 13

4.2 Package alert.client.model . . . . . 13

4.2.1 Detailed Description . . . . . 13

4.3 Package alert.controller . . . . . 14

4.3.1 Detailed Description . . . . . 14

4.4 Package alert.controller.model . . . . . 14

4.4.1 Detailed Description . . . . . 15

4.5 Package alert.domain . . . . . 15

4.5.1 Detailed Description . . . . . 15

4.6 Package alert.domain.model . . . . . 15

4.6.1 Detailed Description . . . . . 16

4.7 Package alert.enums . . . . . 16

4.7.1 Detailed Description . . . . . 16

4.8 Package alert.service . . . . . 16

4.8.1 Detailed Description . . . . . 17

4.9 Package common . . . . . 17

4.9.1 Detailed Description . . . . . 17

4.10 Package tracking . . . . . 18

4.10.1 Detailed Description . . . . . 19

4.11 Package transit . . . . . 19

4.11.1 Detailed Description . . . . . 20

4.12 Package user . . . 20
4.12.1 Detailed Description . . . 21

**5 Class Documentation** **23**

5.1 AbstractFeedParserTemplate Class Reference . . . 23
5.1.1 Detailed Description . . . 24
5.1.2 Member Function Documentation . . . 24
5.1.2.1 loadFeed . . . 24
5.1.2.2 parseFeed . . . 24
5.1.2.3 saveRoutes . . . 25
5.1.2.4 start . . . 25
5.1.2.5 validate . . . 25
5.2 Alert Class Reference . . . 25
5.2.1 Detailed Description . . . 27
5.2.2 Member Data Documentation . . . 27
5.2.2.1 alertInitiator . . . 27
5.2.2.2 alertRunType . . . 27
5.2.2.3 alertType . . . 27
5.2.2.4 Status . . . 27
5.2.2.5 userInformation . . . 27
5.3 AlertExecuteStrategyFactory Class Reference . . . 27
5.3.1 Detailed Description . . . 28
5.3.2 Member Function Documentation . . . 28
5.3.2.1 getAlertExecuteStrategy . . . 28
5.4 AlertExecutionScheduler Class Reference . . . 28
5.4.1 Detailed Description . . . 28
5.4.2 Member Function Documentation . . . 28
5.4.2.1 getAlertServices . . . 28
5.4.2.2 runAlert . . . 29
5.5 AlertFactory Class Reference . . . 29
5.5.1 Detailed Description . . . 29
5.5.2 Member Function Documentation . . . 29
5.5.2.1 createAlert . . . 29
5.5.2.2 createAlert . . . 29
5.6 AlertInitiator Enum Reference . . . 30
5.6.1 Detailed Description . . . 30
5.7 AlertNotificationType Enum Reference . . . 30
5.7.1 Detailed Description . . . 30
5.7.2 Member Data Documentation . . . 30
5.7.2.1 PlannedDisruption . . . 30

5.7.2.2 ScheduleInformation . . . 30
5.7.2.3 UnplannedDisruption . . . 31
5.8 AlertRangeLogic Class Reference . . . 31
5.8.1 Detailed Description . . . 31
5.9 AlertRecurringType Enum Reference . . . 31
5.9.1 Detailed Description . . . 31
5.10 AlertRepository Class Reference . . . 32
5.10.1 Detailed Description . . . 32
5.10.2 Member Function Documentation . . . 32
5.10.2.1 deleteAlert . . . 32
5.10.2.2 getAlertByDateTime . . . 32
5.10.2.3 getAlertByRoute . . . 33
5.10.2.4 getAlertByUserId . . . 33
5.10.2.5 saveAlert . . . 33
5.10.2.6 updateAlert . . . 33
5.11 AlertRequestController Class Reference . . . 34
5.11.1 Detailed Description . . . 34
5.11.2 Member Function Documentation . . . 35
5.11.2.1 processTrackingAlertRequest . . . 35
5.11.2.2 processTransitAlertRequest . . . 35
5.11.2.3 processUserAlertRequest . . . 35
5.11.2.4 verifySession . . . 36
5.12 AlertRequestModel Class Reference . . . 36
5.12.1 Detailed Description . . . 36
5.12.2 Member Data Documentation . . . 37
5.12.2.1 alertInitiator . . . 37
5.13 AlertResponseModel Class Reference . . . 37
5.13.1 Detailed Description . . . 37
5.14 AlertRunType Enum Reference . . . 37
5.14.1 Detailed Description . . . 37
5.15 AlertService Class Reference . . . 37
5.15.1 Detailed Description . . . 38
5.15.2 Member Function Documentation . . . 38
5.15.2.1 createAlert . . . 38
5.15.2.2 deleteAlert . . . 39
5.15.2.3 saveAlert . . . 39
5.15.2.4 sendAlert . . . 39
5.15.2.5 updateAlert . . . 39
5.15.3 Member Data Documentation . . . 40
5.15.3.1 alertExecuteStrategyFactory . . . 40

5.15.3.2 alertFactory . . . 40
5.15.3.3 alertRepository . . . 40
5.16 AlertServiceFactory Class Reference . . . 40
5.16.1 Detailed Description . . . 41
5.16.2 Member Function Documentation . . . 41
5.16.2.1 getAlertService . . . 41
5.16.3 Member Data Documentation . . . 41
5.16.3.1 trackingAlertService . . . 41
5.16.3.2 transitAlertService . . . 41
5.16.3.3 userAlertService . . . 41
5.17 AlertSpecification Interface Reference . . . 41
5.17.1 Detailed Description . . . 42
5.17.2 Member Function Documentation . . . 42
5.17.2.1 inAlertRange . . . 42
5.18 AlertStatus Enum Reference . . . 42
5.18.1 Detailed Description . . . 43
5.18.2 Member Data Documentation . . . 43
5.18.2.1 Deactive . . . 43
5.18.2.2 Error . . . 43
5.18.2.3 Expired . . . 43
5.19 AlertType Enum Reference . . . 43
5.19.1 Detailed Description . . . 43
5.20 AlertUserClient Class Reference . . . 43
5.20.1 Detailed Description . . . 44
5.20.2 Member Function Documentation . . . 44
5.20.2.1 getuserInformation . . . 44
5.20.2.2 verifySessionToken . . . 44
5.21 BaseController Class Reference . . . 44
5.21.1 Detailed Description . . . 45
5.21.2 Member Function Documentation . . . 45
5.21.2.1 handleBusBuddyException . . . 45
5.21.2.2 handleGenericException . . . 45
5.22 BusBuddyBadRequestException Class Reference . . . 46
5.22.1 Detailed Description . . . 46
5.22.2 Member Function Documentation . . . 46
5.22.2.1 getHttpCode . . . 46
5.23 BusBuddyConflictException Class Reference . . . 46
5.23.1 Detailed Description . . . 47
5.23.2 Member Function Documentation . . . 47
5.23.2.1 getHttpCode . . . 47

5.24 BusBuddyException Class Reference . . . 47
5.24.1 Detailed Description . . . 48
5.24.2 Member Function Documentation . . . 48
5.24.2.1 getHttpCode . . . 48
5.25 BusBuddyForbiddenException Class Reference . . . 48
5.25.1 Detailed Description . . . 49
5.25.2 Member Function Documentation . . . 49
5.25.2.1 getHttpCode . . . 49
5.26 BusBuddyInternalException Class Reference . . . 49
5.26.1 Detailed Description . . . 50
5.26.2 Member Function Documentation . . . 50
5.26.2.1 getHttpCode . . . 50
5.27 BusBuddyNotFoundException Class Reference . . . 50
5.27.1 Detailed Description . . . 50
5.27.2 Member Function Documentation . . . 51
5.27.2.1 getHttpCode . . . 51
5.28 BusVehicle Class Reference . . . 51
5.28.1 Detailed Description . . . 51
5.28.2 Member Data Documentation . . . 51
5.28.2.1 alertList . . . 51
5.29 CertificateHandler Class Reference . . . 52
5.29.1 Detailed Description . . . 52
5.29.2 Member Function Documentation . . . 52
5.29.2.1 verifySessionToken . . . 52
5.30 CommercialTracking Class Reference . . . 52
5.30.1 Detailed Description . . . 53
5.30.2 Constructor & Destructor Documentation . . . 53
5.30.2.1 CommercialTracking . . . 53
5.30.3 Member Function Documentation . . . 53
5.30.3.1 getInstance . . . 53
5.31 CommercialTracking.CommercialTrackingHolder Class Reference . . . 54
5.31.1 Detailed Description . . . 54
5.32 DelayAlertLogic Class Reference . . . 54
5.32.1 Detailed Description . . . 54
5.33 Detour Class Reference . . . 54
5.33.1 Detailed Description . . . 55
5.33.2 Constructor & Destructor Documentation . . . 55
5.33.2.1 Detour . . . 55
5.33.3 Member Data Documentation . . . 55
5.33.3.1 cause . . . 55

5.33.3.2 detourId . . . 55
5.33.3.3 estimatedDelay . . . 56
5.34 Fare Class Reference . . . 56
5.34.1 Detailed Description . . . 56
5.34.2 Constructor & Destructor Documentation . . . 56
5.34.2.1 Fare . . . 56
5.34.2.2 Fare . . . 57
5.34.3 Member Function Documentation . . . 57
5.34.3.1 setDiscountedFare . . . 57
5.34.3.2 setRegularFare . . . 57
5.34.4 Member Data Documentation . . . 57
5.34.4.1 discountedFare . . . 57
5.35 FavoriteTransitService Class Reference . . . 57
5.35.1 Detailed Description . . . 58
5.35.2 Constructor & Destructor Documentation . . . 58
5.35.2.1 FavoriteTransitService . . . 58
5.35.3 Member Function Documentation . . . 58
5.35.3.1 getFavoriteRouteIds . . . 58
5.35.3.2 getTransitServiceUrl . . . 58
5.35.3.3 isFavoriteTransitService . . . 58
5.35.3.4 setFavoriteRouteIds . . . 59
5.35.3.5 setFavoriteTransitService . . . 59
5.36 GoogleTransitServiceAdapter Class Reference . . . 59
5.36.1 Detailed Description . . . 59
5.36.2 Constructor & Destructor Documentation . . . 60
5.36.2.1 GoogleTransitServiceAdapter . . . 60
5.36.3 Member Function Documentation . . . 60
5.36.3.1 getRoute . . . 60
5.36.3.2 getRoutes . . . 60
5.37 GoogleTransitServiceAPI Interface Reference . . . 61
5.37.1 Detailed Description . . . 61
5.38 GPSLocationObject Class Reference . . . 61
5.38.1 Detailed Description . . . 61
5.39 GPSLocationObserver Class Reference . . . 61
5.39.1 Detailed Description . . . 62
5.39.2 Member Function Documentation . . . 62
5.39.2.1 getGPSLocation . . . 62
5.39.2.2 gpsUpdate . . . 62
5.39.2.3 setGPSLocation . . . 63
5.40 GPSLocationTracking Class Reference . . . 63

5.40.1 Detailed Description . . . 63
5.40.2 Member Function Documentation . . . 63
5.40.2.1 registerGPSDevice . . . 63
5.40.2.2 unregisterGPSDevice . . . 64
5.41 GPSPuller Class Reference . . . 64
5.41.1 Detailed Description . . . 65
5.41.2 Constructor & Destructor Documentation . . . 65
5.41.2.1 GPSPuller . . . 65
5.42 GPSPuller.GPSPullerHolder Class Reference . . . 65
5.42.1 Detailed Description . . . 65
5.43 GPSPusher Class Reference . . . 65
5.43.1 Detailed Description . . . 66
5.43.2 Constructor & Destructor Documentation . . . 66
5.43.2.1 GPSPusher . . . 66
5.43.3 Member Function Documentation . . . 66
5.43.3.1 getInstance . . . 66
5.44 GPSPusher.GPSPusherHolder Class Reference . . . 67
5.44.1 Detailed Description . . . 67
5.45 GPSVehicleTracker Class Reference . . . 67
5.45.1 Detailed Description . . . 67
5.45.2 Constructor & Destructor Documentation . . . 68
5.45.2.1 GPSVehicleTracker . . . 68
5.46 GTFSFeedParser Class Reference . . . 68
5.46.1 Detailed Description . . . 68
5.46.2 Member Function Documentation . . . 68
5.46.2.1 parseFeed . . . 68
5.47 HashUtility Class Reference . . . 68
5.47.1 Detailed Description . . . 69
5.47.2 Member Function Documentation . . . 69
5.47.2.1 hash . . . 69
5.48 IAlertExecuteStrategy Interface Reference . . . 69
5.48.1 Detailed Description . . . 69
5.48.2 Member Function Documentation . . . 69
5.48.2.1 execute . . . 70
5.49 InvalidRouteParseException Class Reference . . . 70
5.49.1 Detailed Description . . . 70
5.49.2 Constructor & Destructor Documentation . . . 71
5.49.2.1 InvalidRouteParseException . . . 71
5.49.3 Member Data Documentation . . . 71
5.49.3.1 routeBatch . . . 71

5.49.3.2 serialVersionUID . . . 71
5.50 ISessionHandler Interface Reference . . . 71
5.50.1 Detailed Description . . . 71
5.50.2 Member Function Documentation . . . 71
5.50.2.1 verifySessionToken . . . 71
5.51 ITeamTransitServiceController Class Reference . . . 72
5.51.1 Detailed Description . . . 73
5.51.2 Member Function Documentation . . . 73
5.51.2.1 getRoute . . . 73
5.51.2.2 getRoutes . . . 73
5.51.2.3 getServiceURL . . . 74
5.51.2.4 getTransitInfo . . . 74
5.51.2.5 handleRouteDisruptionEvent . . . 74
5.51.3 Member Data Documentation . . . 74
5.51.3.1 alertRequestController . . . 74
5.51.3.2 transitFeed . . . 75
5.52 ITeamTripServiceController Class Reference . . . 75
5.52.1 Detailed Description . . . 75
5.52.2 Member Function Documentation . . . 75
5.52.2.1 calculateTrip . . . 75
5.53 ITeamUserFavoritesService Class Reference . . . 76
5.53.1 Detailed Description . . . 76
5.53.2 Member Function Documentation . . . 76
5.53.2.1 readFavorites . . . 76
5.53.2.2 saveFavorites . . . 76
5.54 ITeamUserLoginService Class Reference . . . 77
5.54.1 Detailed Description . . . 77
5.54.2 Member Function Documentation . . . 77
5.54.2.1 checkPermissions . . . 77
5.54.2.2 createAlertSession . . . 78
5.54.2.3 getUser . . . 78
5.54.2.4 login . . . 78
5.54.2.5 logout . . . 78
5.54.2.6 resetPassword . . . 78
5.54.2.7 resetPassword . . . 79
5.54.2.8 sendUsername . . . 79
5.54.2.9 sendUsername . . . 79
5.55 ITeamUserManagementService Class Reference . . . 79
5.55.1 Detailed Description . . . 79
5.55.2 Member Function Documentation . . . 80

5.55.2.1 createUser . . . 80
5.55.2.2 deleteUser . . . 80
5.55.2.3 findUserByEmail . . . 80
5.55.2.4 findUserByMobile . . . 80
5.55.2.5 findUserByUsername . . . 80
5.55.2.6 updateUser . . . 80
5.56 ITrackingService Interface Reference . . . 80
5.56.1 Detailed Description . . . 81
5.56.2 Member Function Documentation . . . 81
5.56.2.1 addUserTrackingAlert . . . 81
5.56.2.2 getTransitVehicleLocation . . . 81
5.56.2.3 registerVehicleOnRoute . . . 82
5.56.2.4 startTrackingController . . . 82
5.56.2.5 unregisterVehicleFromRoute . . . 82
5.57 Location Class Reference . . . 82
5.57.1 Detailed Description . . . 82
5.57.2 Constructor & Destructor Documentation . . . 83
5.57.2.1 Location . . . 83
5.58 MessageDeliveryUtility Class Reference . . . 83
5.58.1 Detailed Description . . . 83
5.58.2 Member Function Documentation . . . 83
5.58.2.1 sendEmail . . . 83
5.58.2.2 sendSms . . . 83
5.59 OneTimeAlert Class Reference . . . 84
5.59.1 Detailed Description . . . 84
5.59.2 Member Data Documentation . . . 84
5.59.2.1 dateExecuted . . . 84
5.60 PersistedTransitFeed Class Reference . . . 85
5.60.1 Detailed Description . . . 85
5.60.2 Member Function Documentation . . . 85
5.60.2.1 getRoute . . . 85
5.60.2.2 getRoutes . . . 86
5.61 RecurringAlert Class Reference . . . 86
5.61.1 Detailed Description . . . 87
5.61.2 Member Data Documentation . . . 87
5.61.2.1 alertRecurringType . . . 87
5.61.2.2 repeatEvery . . . 87
5.62 RecurringData Class Reference . . . 87
5.62.1 Detailed Description . . . 88
5.62.2 Member Data Documentation . . . 88

5.62.2.1 dayOfMonth . . . 88
5.62.2.2 dayOfWeek . . . 88
5.62.2.3 dayOfYear . . . 88
5.62.2.4 startHour . . . 89
5.62.2.5 startMinute . . . 89
5.63 Route Class Reference . . . 89
5.63.1 Detailed Description . . . 89
5.63.2 Constructor & Destructor Documentation . . . 90
5.63.2.1 Route . . . 90
5.63.3 Member Function Documentation . . . 90
5.63.3.1 setStops . . . 90
5.63.4 Member Data Documentation . . . 90
5.63.4.1 detours . . . 90
5.63.4.2 routeName . . . 90
5.63.4.3 stops . . . 90
5.64 RouteAlertExecuteStrategy Class Reference . . . 90
5.64.1 Detailed Description . . . 91
5.64.2 Member Function Documentation . . . 91
5.64.2.1 execute . . . 91
5.64.3 Member Data Documentation . . . 91
5.64.3.1 alertRepository . . . 91
5.64.3.2 userClient . . . 91
5.65 RouteDisruptionAlert Class Reference . . . 92
5.65.1 Detailed Description . . . 92
5.65.2 Member Data Documentation . . . 92
5.65.2.1 routeId . . . 92
5.65.2.2 transitServiceUrl . . . 92
5.66 RouteDisruptionEvent Class Reference . . . 92
5.66.1 Detailed Description . . . 93
5.66.2 Constructor & Destructor Documentation . . . 93
5.66.2.1 RouteDisruptionEvent . . . 93
5.67 RouteRepository Interface Reference . . . 93
5.67.1 Detailed Description . . . 93
5.67.2 Member Function Documentation . . . 94
5.67.2.1 delete . . . 94
5.67.2.2 getAll . . . 94
5.67.2.3 read . . . 94
5.67.2.4 save . . . 94
5.67.2.5 save . . . 95
5.68 RouteSpecification Class Reference . . . 95

5.68.1 Detailed Description . . . 95
5.68.2 Member Function Documentation . . . 96
5.68.2.1 isSatisfiedBy . . . 96
5.69 ScheduleAlertExecuteStrategy Class Reference . . . 96
5.69.1 Detailed Description . . . 97
5.69.2 Member Data Documentation . . . 97
5.69.2.1 alertRepository . . . 97
5.69.2.2 userClient . . . 97
5.70 Session Class Reference . . . 97
5.70.1 Detailed Description . . . 98
5.70.2 Constructor & Destructor Documentation . . . 98
5.70.2.1 Session . . . 98
5.70.3 Member Function Documentation . . . 98
5.70.3.1 getCreationTime . . . 98
5.70.3.2 getExpirationTime . . . 98
5.70.3.3 getSessionToken . . . 98
5.70.3.4 getUserId . . . 99
5.70.3.5 isAlertSession . . . 99
5.70.3.6 isValid . . . 99
5.70.3.7 setExpirationTime . . . 99
5.70.3.8 setValid . . . 99
5.71 SessionRepository Class Reference . . . 99
5.71.1 Detailed Description . . . 100
5.71.2 Member Function Documentation . . . 100
5.71.2.1 createSession . . . 100
5.71.2.2 getSession . . . 100
5.71.2.3 killSession . . . 101
5.72 SessionTokenHandler Class Reference . . . 101
5.72.1 Detailed Description . . . 102
5.72.2 Member Function Documentation . . . 102
5.72.2.1 verifySessionToken . . . 102
5.72.3 Member Data Documentation . . . 102
5.72.3.1 userClient . . . 102
5.73 SessionVerificationFactory Class Reference . . . 102
5.73.1 Detailed Description . . . 103
5.73.2 Member Function Documentation . . . 103
5.73.2.1 getSessionTokenVerificationStrategy . . . 103
5.74 Specification< T > Interface Reference . . . 103
5.74.1 Detailed Description . . . 103
5.74.2 Member Function Documentation . . . 103

5.74.2.1 and . . . 103
5.74.2.2 isSatisfiedBy . . . 104
5.74.2.3 not . . . 104
5.74.2.4 or . . . 104
5.75 Stop Class Reference . . . 104
5.75.1 Detailed Description . . . 105
5.75.2 Constructor & Destructor Documentation . . . 105
5.75.2.1 Stop . . . 105
5.75.3 Member Function Documentation . . . 105
5.75.3.1 getStopTimes . . . 105
5.75.4 Member Data Documentation . . . 106
5.75.4.1 covered . . . 106
5.75.4.2 description . . . 106
5.75.4.3 stopId . . . 106
5.76 TrackingAlertFactory Class Reference . . . 106
5.76.1 Detailed Description . . . 106
5.76.2 Member Function Documentation . . . 107
5.76.2.1 createAlertObserver . . . 107
5.77 TrackingAlertObserver Class Reference . . . 107
5.77.1 Detailed Description . . . 107
5.77.2 Member Function Documentation . . . 108
5.77.2.1 getSpec . . . 108
5.77.2.2 setSpec . . . 108
5.77.2.3 updateAlert . . . 108
5.78 TrackingAlertRequestModel Class Reference . . . 108
5.78.1 Detailed Description . . . 108
5.79 TrackingAlertService Class Reference . . . 109
5.79.1 Detailed Description . . . 109
5.79.2 Member Function Documentation . . . 109
5.79.2.1 createAlert . . . 109
5.79.2.2 sendAlert . . . 109
5.79.2.3 updateAlert . . . 110
5.80 TrackingDelayAlert Class Reference . . . 110
5.80.1 Detailed Description . . . 110
5.80.2 Member Function Documentation . . . 110
5.80.2.1 updateAlert . . . 110
5.81 TrackingLocationAlert Class Reference . . . 111
5.81.1 Detailed Description . . . 111
5.81.2 Constructor & Destructor Documentation . . . 111
5.81.2.1 TrackingLocationAlert . . . 111

5.82 TrackingResponseModel Class Reference . . . 111
5.82.1 Detailed Description . . . 112
5.82.2 Member Function Documentation . . . 112
5.82.2.1 convertJSONAlertInput . . . 112
5.82.2.2 convertJSONVehicleInput . . . 112
5.82.2.3 formatJSONResponse . . . 112
5.83 TrackingServiceController Class Reference . . . 112
5.83.1 Detailed Description . . . 113
5.83.2 Member Function Documentation . . . 113
5.83.2.1 addUserTrackingAlert . . . 113
5.83.2.2 registerVehicleOnRoute . . . 113
5.83.2.3 unregisterVehicleFromRoute . . . 114
5.84 TransitAlertRequestModel Class Reference . . . 114
5.84.1 Detailed Description . . . 114
5.85 TransitAlertService Class Reference . . . 114
5.85.1 Detailed Description . . . 115
5.85.2 Member Function Documentation . . . 115
5.85.2.1 createAlert . . . 115
5.85.2.2 sendAlert . . . 115
5.85.2.3 updateAlert . . . 115
5.86 TransitFeed Interface Reference . . . 115
5.86.1 Detailed Description . . . 116
5.86.2 Member Function Documentation . . . 116
5.86.2.1 getRoute . . . 116
5.86.2.2 getRoutes . . . 116
5.87 TransitInfo Class Reference . . . 117
5.87.1 Detailed Description . . . 117
5.87.2 Member Data Documentation . . . 117
5.87.2.1 logo . . . 117
5.87.2.2 transitAuthorityName . . . 118
5.87.2.3 website . . . 118
5.88 TransitProvider Class Reference . . . 118
5.88.1 Detailed Description . . . 118
5.88.2 Member Function Documentation . . . 118
5.88.2.1 fireRouteDistruptionEvent . . . 118
5.88.2.2 registerObserver . . . 119
5.88.2.3 unregisterObserver . . . 119
5.88.3 Member Data Documentation . . . 119
5.88.3.1 name . . . 119
5.88.3.2 providerId . . . 119

5.89 TransitProviderObserver Interface Reference . . . 119
5.89.1 Detailed Description . . . 120
5.89.2 Member Function Documentation . . . 120
5.89.2.1 handleRouteDisruptionEvent . . . 120
5.90 TransitService Interface Reference . . . 120
5.90.1 Detailed Description . . . 120
5.90.2 Member Function Documentation . . . 120
5.90.2.1 getRoute . . . 120
5.90.2.2 getRoutes . . . 121
5.90.2.3 getServiceURL . . . 121
5.90.2.4 getTransitInfo . . . 122
5.91 TransitVehicle Class Reference . . . 122
5.91.1 Detailed Description . . . 122
5.91.2 Member Function Documentation . . . 123
5.91.2.1 addAlertSpecification . . . 123
5.91.2.2 checkForAlerts . . . 123
5.92 TransitVehicleFactory Class Reference . . . 123
5.92.1 Detailed Description . . . 123
5.92.2 Member Function Documentation . . . 123
5.92.2.1 createTransitVehicle . . . 123
5.92.2.2 getGPSTypeFromURL . . . 124
5.92.2.3 getVehicleGPSDeviceID . . . 124
5.93 Trip Class Reference . . . 124
5.93.1 Detailed Description . . . 124
5.93.2 Constructor & Destructor Documentation . . . 125
5.93.2.1 Trip . . . 125
5.94 TripInformation Class Reference . . . 125
5.94.1 Detailed Description . . . 125
5.94.2 Member Function Documentation . . . 125
5.94.2.1 getRouteIds . . . 125
5.94.3 Member Data Documentation . . . 126
5.94.3.1 tripData . . . 126
5.95 TripService Interface Reference . . . 126
5.95.1 Detailed Description . . . 126
5.95.2 Member Function Documentation . . . 126
5.95.2.1 calculateTrip . . . 126
5.96 User Class Reference . . . 127
5.96.1 Detailed Description . . . 128
5.96.2 Constructor & Destructor Documentation . . . 128
5.96.2.1 User . . . 128

5.96.3 Member Function Documentation . . . 128
5.96.3.1 getCountryCode . . . 128
5.96.3.2 getEmail . . . 128
5.96.3.3 getFirstName . . . 128
5.96.3.4 getMobile . . . 129
5.96.3.5 getPasswordHash . . . 129
5.96.3.6 getUserId . . . 129
5.96.3.7 getUsername . . . 129
5.96.3.8 getUserType . . . 129
5.96.3.9 isForcePasswordChange . . . 130
5.96.3.10 setCountryCode . . . 130
5.96.3.11 setEmail . . . 130
5.96.3.12 setFirstName . . . 130
5.96.3.13 setForcePasswordChange . . . 130
5.96.3.14 setMobile . . . 130
5.96.3.15 setPasswordHash . . . 131
5.96.3.16 setUserType . . . 131
5.97 UserAlertRequestModel Class Reference . . . 131
5.97.1 Detailed Description . . . 131
5.98 UserAlertService Class Reference . . . 131
5.98.1 Detailed Description . . . 132
5.98.2 Member Function Documentation . . . 132
5.98.2.1 createAlert . . . 132
5.98.2.2 sendAlert . . . 132
5.98.2.3 updateAlert . . . 132
5.99 UserFavoritesList Class Reference . . . 133
5.99.1 Detailed Description . . . 133
5.99.2 Constructor & Destructor Documentation . . . 133
5.99.2.1 UserFavoritesList . . . 133
5.99.3 Member Function Documentation . . . 133
5.99.3.1 getFavoriteTransitServices . . . 133
5.99.3.2 getUserId . . . 133
5.99.3.3 setFavoriteTransitServices . . . 134
5.100 UserFavoritesRepository Class Reference . . . 134
5.100.1 Detailed Description . . . 134
5.100.2 Member Function Documentation . . . 134
5.100.2.1 getFavorites . . . 134
5.100.2.2 updateFavorites . . . 135
5.101 UserFavoritesService Interface Reference . . . 135
5.101.1 Detailed Description . . . 135

5.101.2 Member Function Documentation . . . 135
5.101.2.1 readFavorites . . . 136
5.101.2.2 saveFavorites . . . 136
5.102 UserInformation Class Reference . . . 136
5.102.1 Detailed Description . . . 137
5.103 UserLoginService Interface Reference . . . 137
5.103.1 Detailed Description . . . 138
5.103.2 Member Function Documentation . . . 138
5.103.2.1 createAlertSession . . . 138
5.103.2.2 getUser . . . 139
5.103.2.3 login . . . 139
5.103.2.4 logout . . . 140
5.103.2.5 resetPassword . . . 140
5.103.2.6 resetPassword . . . 141
5.103.2.7 sendUsername . . . 141
5.103.2.8 sendUsername . . . 142
5.104 UserManagementService Interface Reference . . . 142
5.104.1 Detailed Description . . . 143
5.104.2 Member Function Documentation . . . 143
5.104.2.1 createUser . . . 143
5.104.2.2 deleteUser . . . 144
5.104.2.3 findUserByEmail . . . 144
5.104.2.4 findUserByMobile . . . 145
5.104.2.5 findUserByUsername . . . 146
5.104.2.6 updateUser . . . 146
5.105 UserRepository Class Reference . . . 147
5.105.1 Detailed Description . . . 147
5.105.2 Member Function Documentation . . . 147
5.105.2.1 createUser . . . 147
5.105.2.2 deleteUser . . . 148
5.105.2.3 getUserByEmail . . . 148
5.105.2.4 getUserById . . . 149
5.105.2.5 getUserByMobile . . . 149
5.105.2.6 getUserByUsername . . . 150
5.105.2.7 updateUser . . . 151
5.106 UserSessionInformation Class Reference . . . 151
5.106.1 Detailed Description . . . 151
5.106.2 Member Data Documentation . . . 152
5.106.2.1 userSessionToken . . . 152
5.107 UserTrackingAlertObject Class Reference . . . 152

5.107.1 Detailed Description . . . 152
5.107.2 Member Data Documentation . . . 152
5.107.2.1 alertTime . . . 152
5.107.2.2 routeID . . . 152
5.107.2.3 scheduledTime . . . 153
5.107.2.4 stopLocation . . . 153
5.107.2.5 transitCoInfo . . . 153
5.107.2.6 type . . . 153
5.107.2.7 userContactInfo . . . 153
5.108 UserType Enum Reference . . . 153
5.108.1 Detailed Description . . . 153
5.109 VehicleObject Class Reference . . . 154
5.109.1 Detailed Description . . . 154
5.109.2 Member Data Documentation . . . 154
5.109.2.1 currentRoute . . . 154
5.109.2.2 gpsDeviceID . . . 154
5.109.2.3 gpsDeviceInfo . . . 154
5.109.2.4 transitCoURL . . . 154
5.110 VehicleRepository Class Reference . . . 155
5.110.1 Detailed Description . . . 155
5.110.2 Member Function Documentation . . . 155
5.110.2.1 findVehicle . . . 155
5.110.2.2 findVehiclesByRoute . . . 155
5.110.2.3 removeVehicle . . . 156
5.110.2.4 updateVehicle . . . 156

**Index** **156**

# Chapter 1

# Namespace Index

## 1.1 Packages

Here are the packages with brief descriptions (if available):


alert.client
: This package contains client layer classes that facilitates the call to other modules.mWe use client layer to communicate and gather information from external system . . . . . 13

alert.client.model
: This package contains model needed for client layer classes . . . . . 13

alert.controller
: This package contains controller layer classes for Alert module which takes in REST request from external sources . . . . . 14

alert.controller.model
: This package contains model needed for controller layer . . . . . 14

alert.domain
: This package contains domain layer classes for Alert Module . . . . . 15

alert.domain.model
: This package contains entity and aggregate needed for domain layer in Alert Module . . . . . 15

alert.enums
: This package contains enums needed for Alert module . . . . . 16

alert.service
: This package contains service layer classes needed for Alert Module . . . . . 16

common
: This package contains common BusBuddy objects and utilities to be used by all modules . . . 17

tracking
: Tracking Module manages GPS vehicle tracking and alerting subscribed users based on vehicle location . . . . . 18

transit
: The Transit Module is an interface to get Route/Fare/Detour information from a TransitProvider 19

user
: This package contains the objects used by the User Module of the BusBuddy application . . . 20

# Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:


AbstractFeedParserTemplate . . . 23
- GTFSFeedParser . . . 68

AlertExecuteStrategyFactory . . . 27
AlertExecutionScheduler . . . 28
AlertFactory . . . 29
AlertInitiator . . . 30
AlertNotificationType . . . 30
AlertRecurringType . . . 31
AlertRepository . . . 32
AlertRequestController . . . 34
AlertRequestModel . . . 36
- TrackingAlertRequestModel . . . 108
- TransitAlertRequestModel . . . 114
- UserAlertRequestModel . . . 131

AlertResponseModel . . . 37
AlertRunType . . . 37
AlertService . . . 37
- TrackingAlertService . . . 109
- TransitAlertService . . . 114
- UserAlertService . . . 131

AlertServiceFactory . . . 40
AlertSpecification . . . 41
- AlertRangeLogic . . . 31
- DelayAlertLogic . . . 54

AlertStatus . . . 42
AlertType . . . 43
AlertUserClient . . . 43
BaseController . . . 44
- ITeamTransitServiceController . . . 72
- ITeamTripServiceController . . . 75

CommercialTracking.CommercialTrackingHolder . . . 54
Detour . . . 54
Exception
- BusBuddyException . . . 47
  - BusBuddyBadRequestException . . . 46
  - BusBuddyConflictException . . . 46

BusBuddyForbiddenException . . . 48
BusBuddyInternalException . . . 49
BusBuddyNotFoundException . . . 50
InvalidRouteParseException . . . 70
Fare . . . 56
FavoriteTransitService . . . 57
GoogleTransitServiceAPI . . . 61
GPSLocationObject . . . 61
GPSLocationObserver . . . 61
GPSVehicleTracker . . . 67
GPSLocationTracking . . . 63
CommercialTracking . . . 52
GPSPuller . . . 64
GPSPusher . . . 65
GPSPuller.GPSPullerHolder . . . 65
GPSPusher.GPSPusherHolder . . . 67
HashUtility . . . 68
IAlertExecuteStrategy . . . 69
RouteAlertExecuteStrategy . . . 90
ScheduleAlertExecuteStrategy . . . 96
ISessionHandler . . . 71
CertificateHandler . . . 52
SessionTokenHandler . . . 101
ITrackingService . . . 80
TrackingServiceController . . . 112
Location . . . 82
MessageDeliveryUtility . . . 83
RecurringData . . . 87
Route . . . 89
RouteDisruptionAlert . . . 92
RouteDisruptionEvent . . . 92
RouteRepository . . . 93
Session . . . 97
SessionRepository . . . 99
SessionVerificationFactory . . . 102
Specification< T > . . . 103
Stop . . . 104
TrackingAlertFactory . . . 106
TrackingAlertObserver . . . 107
TrackingDelayAlert . . . 110
TrackingLocationAlert . . . 111
TrackingResponseModel . . . 111
TransitFeed . . . 115
GoogleTransitServiceAdapter . . . 59
PersistedTransitFeed . . . 85
TransitInfo . . . 117
TransitProvider . . . 118
TransitProviderObserver . . . 119
ITeamTransitServiceController . . . 72
TransitService . . . 120
ITeamTransitServiceController . . . 72
TransitVehicle . . . 122
BusVehicle . . . 51
TransitVehicleFactory . . . 123
Trip . . . 124

TripInformation . . . . . 125
TripService . . . . . 126
ITeamTripServiceController . . . . . 75
User . . . . . 127
UserFavoritesList . . . . . 133
UserFavoritesRepository . . . . . 134
UserFavoritesService . . . . . 135
ITeamUserFavoritesService . . . . . 76
UserInformation . . . . . 136
UserLoginService . . . . . 137
ITeamUserLoginService . . . . . 77
UserManagementService . . . . . 142
ITeamUserManagementService . . . . . 79
UserRepository . . . . . 147
UserSessionInformation . . . . . 151
UserTrackingAlertObject . . . . . 152
UserType . . . . . 153
VehicleObject . . . . . 154
VehicleRepository . . . . . 155
Serializable
Alert . . . . . 25
OneTimeAlert . . . . . 84
RecurringAlert . . . . . 86
Specification
RouteSpecification . . . . . 95

# Chapter 3

# Class Index

## 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:


AbstractFeedParserTemplate
: A Template Method pattern to allow for the import of data from different TransitProviders in potentially different formats . . . 23

Alert
: This is a base Alert Model that has most of the common information about an Alert . . . 25

AlertExecuteStrategyFactory
: This factory is going to pick the best available strategy to execute the alert so that the user can be notified . . . 27

AlertExecutionScheduler
: This is a background service that runs all the time . . . 28

AlertFactory
: An alert factory that can create different types of semi-populated alert models depending upon the information . . . 29

AlertInitiator
: A list of enums corresponding with modules that initiates the call . . . 30

AlertNotificationType
: Notification type of alert, . . . 30

AlertRangeLogic
: Alert Range Logic implements the business logic to determine if a vehicle is within a range where an alert needs to be sent to a user who has registered for tracking alerts . . . 31

AlertRecurringType
: Represents different recurring type of alert i.e., Yearly, Monthly, Weekly or Daily . . . 31

AlertRepository
: A Repository that handles the persistent behavior of the Alert aggregate . . . 32

AlertRequestController
: This is a front facing controller that takes in request from other module via REST call and returns a JSON representation data . . . 34

AlertRequestModel
: A base model that contains bare minimum information about the alert request . . . 36

AlertResponseModel
: This is a basic alert response model that is returned for every alert related requested . . . 37

AlertRunType
: Represents the type of alert depending upon the run type whether to run once or to run in a fixed recurring manner . . . 37

AlertService
: Alert Service is a base class that is extended by other module specific services . . . 37

AlertServiceFactory
AlertServiceFactory initializes appropriate alert service depending upon the parameter being passed . . . 40
AlertSpecification
Interface for Alert Specifications which contain the business logic used to determine if an alert should be triggered for a vehicle . . . 41
AlertStatus
Represents the status of an alert depending upon its state in its life cycle . . . 42
AlertType
Enumeration of the alert types recognized by bus buddy . . . 43
AlertUserClient
Client layer will handle all the responsibility of sending request to other modules or external sources . . . 43
BaseController
This is a base class to be extended by each of the controller classes . . . 44
BusBuddyBadRequestException
This exception object represents internal errors which may occur as a result of an error in the client's request . . . 46
BusBuddyConflictException
This exception object is thrown when a request would create a conflict which violates constraints set within the system . . . 46
BusBuddyException
This exception object is an abstract base class . . . 47
BusBuddyForbiddenException
This exception object represents internal errors which may occur as a result of attempts to access a resource without authorization . . . 48
BusBuddyInternalException
This exception object represents internal errors which may occur, which are generally not due to the specifics of what appears to be a valid request . . . 49
BusBuddyNotFoundException
This exception object represents the error that occurs when a resource cannot be found . . . 50
BusVehicle
Bus Vehicle is a concrete implementation of the abstract Transit Vehicle . . . 51
CertificateHandler
A concrete strategy implementation of ISessionHandler that can verify the certificate token being passed . . . 52
CommercialTracking
Implements Subject GPSLocationTracking for retrieving GPS location updates from outside commercial tracking services . . . 52
CommercialTracking.CommercialTrackingHolder
Commercial Tracking Holder is loaded on the first execution of CommercialTracking.getInstance() or the first access to CommercialTracking.INSTANCE, not before (lazy instantiation) . . . 54
DelayAlertLogic
Implements the business logic to determine if the vehicle is behind schedule or not reporting GPS updates and send a notification to the transit company . . . 54
Detour
A disruption in service due to an unexpected event . . . 54
Fare
An immutable Value Object representing the cost, or 'fare,' required to ride a TransitVehicle on a particular Route . . . 56
FavoriteTransitService
This class is a single transit service in a user's list of favorites . . . 57
GoogleTransitServiceAdapter
An Adapter Class to allow a GoogleTransitServiceAPI service to appear as a TransitService . . . 59
GoogleTransitServiceAPI
A client to Google's Maps API . . . 61
GPSLocationObject
GPS Location is a value object used for GPS coordinates and the time of the last update . . . 61

GPSLocationObserver
Observer Pattern - Observer interface for GPS location tracking . . . 61
GPSLocationTracking
GPSLocationTracking - interface Subject of the Observer Pattern Defines methods for an observer GPS Device to register and receive updates on vehicle location . . . 63
GPSPuller
GPS Puller is a concrete implementation of GPSLocationTracking for obtaining coordinates directly from a GPS device installed in a registered vehicle . . . 64
GPSPuller.GPSPullerHolder
GPS Puller Holder is loaded on the first execution of GPSPuller.getInstance() or the first access to GPSPuller.INSTANCE, not before (lazy instantiation) . . . 65
GPSPusher
Implements Subject GPSLocationTracking for retrieving GPS location updates from registered vehicles . . . 65
GPSPusher.GPSPusherHolder
GPS Pusher Holder is loaded on the first execution of GPSPusher.getInstance() or the first access to GPSPusher.INSTANCE, not before (lazy instantiation) . . . 67
GPSVehicleTracker
Implementation of the Observer, update transit vehicle GPS location and time GPS Vehicle Tracker gets the state as new GPS coordinates and time from GPS Location Tracking and updates the transit vehicle . . . 67
GTFSFeedParser
A AbstractFeedParserTemplate implementation designed to parse GTFS format ZIP files into Routes . . . 68
HashUtility
This is a utility class to handle secure hashes . . . 68
IAlertExecuteStrategy
An interface for executing different type of alert based on alert type . . . 69
InvalidRouteParseException
An InvalidRouteParseException indicates an invalid batch of parsed Routes has been been detected . . . 70
ISessionHandler
An interface to verify the validity of encrypted token being passed . . . 71
ITeamTransitServiceController
The iTeam implementation of the TransitService that exposes Transit data via a REST Service 72
ITeamTripServiceController
An iTeam implementation of the TripService that exposes Trip data via a REST Service . . . 75
ITeamUserFavoritesService
This is the iTeam's implementation of UserFavoritesService . . . 76
ITeamUserLoginService
This is the iTeam's implementation of UserLoginService . . . 77
ITeamUserManagementService
This is the iTeam's implementation of UserManagementService . . . 79
ITrackingService
Interface for the Tracking Service Controller . . . 80
Location
An immutable Value Object representing a physical point on the geographic coordinate system 82
MessageDeliveryUtility
This is a utility class to handle message delivery, such as through email or SMS . . . 83
OneTimeAlert
This is a model of alert that is to be run one time only . . . 84
PersistedTransitFeed
An implementation of the TransitFeed interface that communicates with a RouteRepository to retrieve its data . . . 85
RecurringAlert
This is a model of alert that is to be run multiple times . . . 86
RecurringData
This model holds the information about the date and time the alert needs to run . . . 87

Route
A Route is a TransitVehicle path of travel, or a "Line," as referred to by a TransitProvider . . . . 89
RouteAlertExecuteStrategy
A concrete implementation of IAlertExecuteStrategy that handles executing alert related to route changes . . . . 90
RouteDisruptionAlert
An Alert indicating a disruption of normal Route availability or scheduling . . . . 92
RouteDisruptionEvent
An event indicating a disruption in normal Route scheduling or service . . . . 92
RouteRepository
A Repository Pattern supporting lifecycle operations of Routes, such as Read, Save, Delete, and Query functionality . . . . 93
RouteSpecification
A Specification Pattern class for validating a Route . . . . 95
ScheduleAlertExecuteStrategy
A concrete implementation of IAlertExecuteStrategy that handles executing alert related to user's regular schedule . . . . 96
Session
This class represents a single session for a user of the system, and all of the state data associated with that session . . . . 97
SessionRepository
This class is responsible for handling database access for Sessions, and to construct, persist, and retrieve Session objects . . . . 99
SessionTokenHandler
A concrete strategy implementation of ISessionHandler that can verify the session token being passed . . . . 101
SessionVerificationFactory
A factory that picks the appropriate strategy ISessionHandler to verify the encrypted token . . 102
Specification< T >
A Generic Specification to be used for chaining business validation rules together . . . . 103
Stop
A point on a Route in which a TransitVehicle will stop to pick up and drop off passengers . . . . 104
TrackingAlertFactory
The Alert Factory handles the creation of a user alert . . . . 106
TrackingAlertObserver
Abstract class defining the methods for the tracking alert observer . . . . 107
TrackingAlertRequestModel
This model is a JSON representation of the request from Tracking module . . . . 108
TrackingAlertService
Extends AlertService and provides CRUD methods to operate on alerts as well as sends alerts to execute via appropriate strategy . . . . 109
TrackingDelayAlert
Tracking Alert Observer implements the abstract tracking alert observer and provides the method to actually send an alert to a registered user that their bus is approaching their stop . . . . 110
TrackingLocationAlert
Concrete implementation of the tracking alert observer . . . . 111
TrackingResponseModel
This is a basic tracking response model that is returned for every tracking related request . . . 111
TrackingServiceController
Tracking service controller is the concrete implementation of the tracking service interface . . . 112
TransitAlertRequestModel
This model is a JSON representation of the request from Transit module . . . . 114
TransitAlertService
Extends AlertService and provides CRUD methods to operate on alerts as well as sends alerts to execute via appropriate strategy . . . . 114
TransitFeed
A TransitFeed is an abstraction over a service or set of services that provide information about Routes . . . . 115

TransitInfo
An immutable Value Object describing metadata about a TransitService . . . 117
TransitProvider
A TransitProvider is a description of a company or organization that is the producer of public transportation services . . . 118
TransitProviderObserver
An asynchronous update interface for receiving notifications about TransitProvider Route disruptions . . . 119
TransitService
The TransitService is an interface to get Route/Fare/Detour information from a TransitProvider . 120
TransitVehicle
Abstract transit vehicle class contains the common data for all types of vehicles and the Subject GPS Tracking and the GPS observer to receive GPS location updates . . . 122
TransitVehicleFactory
Transit Vehicle Factory encapsulates the complexity of creating a new vehicle . . . 123
Trip
A Trip is considered an ordered collection of Routes going from a starting point to an ending point 124
TripInformation
This model stores the information about a trip as a value object . . . 125
TripService
A Service to calculate a collection of Routes, or a Trip, allowing for a continuous transit path from a start Location to an end Location . . . 126
User
This class represents a single user of the system, and all of the state data associated with that user . . . 127
UserAlertRequestModel
This model is a JSON representation of the request from User module . . . 131
UserAlertService
Extends AlertService and provides CRUD methods to operate on alerts as well as sends alerts to execute via appropriate strategy . . . 131
UserFavoritesList
This class ties a userId to that user's ordered list of favorites . . . 133
UserFavoritesRepository
This class is responsible for handling database access for favorites, and to persist and retrieve UserFavoritesList objects . . . 134
UserFavoritesService
This is the generic BusBuddy UserFavoritesService interface . . . 135
UserInformation
UserInformation contains the user related data that we get from User Module . . . 136
UserLoginService
This is the generic BusBuddy UserLoginService interface . . . 137
UserManagementService
This is the generic BusBuddy UserManagementService interface . . . 142
UserRepository
This class is responsible for handling database access for User objects, and to construct, persist, and retrieve User objects . . . 147
UserSessionInformation
A model that stores all the information needed to call user module about user information . . . 151
UserTrackingAlertObject
User tracking alert information obtained from the user interface when the user registers for an alert . . . 152
UserType
This is an enumeration of the different statuses that a user can be assigned . . . 153
VehicleObject
Value Object containing vehicle information obtained when the user registers a vehicle using the user interface . . . 154
VehicleRepository
Repository for information on vehicles registered on a route . . . 155

# Chapter 4

# Namespace Documentation

## 4.1 Package alert.client

This package contains client layer classes that facilitates the call to other modules.mWe use client layer to communicate and gather information from external system.

### Packages

- package model

  *This package contains model needed for client layer classes.*

### Classes

- class AlertUserClient

  *Client layer will handle all the responsibility of sending request to other modules or external sources.*

### 4.1.1 Detailed Description

This package contains client layer classes that facilitates the call to other modules.mWe use client layer to communicate and gather information from external system. This layer contains method that can make REST call to other modules and get back appropriate response. Currently, it is only making REST calls, but we can add new classes to make SOAP or any other type of external request. Hence, it abstracts the interaction with outer system and separates that concern to itself.

## 4.2 Package alert.client.model

This package contains model needed for client layer classes.

### Classes

- class UserInformation

  *UserInformation contains the user related data that we get from User Module.*

### 4.2.1 Detailed Description

This package contains model needed for client layer classes.

## 4.3 Package alert.controller

This package contains controller layer classes for Alert module which takes in REST request from external sources.

### Packages

- package model

  *This package contains model needed for controller layer.*

### Classes

- class AlertRequestController

  *This is a front facing controller that takes in request from other module via REST call and returns a JSON representation data.*
- class CertificateHandler

  *A concrete strategy implementation of ISessionHandler that can verify the certificate token being passed.*
- interface ISessionHandler

  *An interface to verify the validity of encrypted token being passed.*
- class SessionTokenHandler

  *A concrete strategy implementation of ISessionHandler that can verify the session token being passed.*
- class SessionVerificationFactory

  *A factory that picks the appropriate strategy ISessionHandler to verify the encrypted token.*

#### 4.3.1 Detailed Description

This package contains controller layer classes for Alert module which takes in REST request from external sources. Controller acts like a facade to most of the modules as the methods are mapped to a certain URI. Other modules can call the REST interface and controller will handle the request. Also, controller handles the session verification depending upon type of encrypted token passed and provides the authorization and authentication. This is achieved by using factory and strategy pattern.

## 4.4 Package alert.controller.model

This package contains model needed for controller layer.

### Classes

- class AlertRequestModel

  *A base model that contains bare minimum information about the alert request.*
- class AlertResponseModel

  *This is a basic alert response model that is returned for every alert related requested.*
- class TrackingAlertRequestModel

  *This model is a JSON representation of the request from Tracking module.*
- class TransitAlertRequestModel

  *This model is a JSON representation of the request from Transit module.*
- class UserAlertRequestModel

  *This model is a JSON representation of the request from User module.*

### 4.4.1 Detailed Description

This package contains model needed for controller layer.

## 4.5 Package alert.domain

This package contains domain layer classes for Alert Module.

### Packages

- package model

  *This package contains entity and aggregate needed for domain layer in Alert Module.*

### Classes

- class AlertFactory

  *An alert factory that can create different types of semi-populated alert models depending upon the information.*
- class AlertRepository

  *A Repository that handles the persistent behavior of the Alert aggregate.*

### 4.5.1 Detailed Description

This package contains domain layer classes for Alert Module. Domain layer classes contains factory and repository that are responsible for handling the models lifecycle. Factory pattern is used to create different types of alert depending upon the parameter.

## 4.6 Package alert.domain.model

This package contains entity and aggregate needed for domain layer in Alert Module.

### Classes

- class Alert

  *This is a base Alert Model that has most of the common information about an Alert.*
- class OneTimeAlert

  *This is a model of alert that is to be run one time only.*
- class RecurringAlert

  *This is a model of alert that is to be run multiple times.*
- class RecurringData

  *This model holds the information about the date and time the alert needs to run.*
- class TripInformation

  *This model stores the information about a trip as a value object.*
- class UserSessionInformation

  *A model that stores all the information needed to call user module about user information.*

### 4.6.1 Detailed Description

This package contains entity and aggregate needed for domain layer in Alert Module. Domain layer models are database representation of the alert model. It contains a root aggregate through which we can enforce business rules and get access to information about its child objects. In the future, if we have to add more fields for alert we can probably implement bridge pattern to provide more flexible structure.

## 4.7 Package alert.enums

This package contains enums needed for Alert module.

### Classes

- enum AlertInitiator

  *A list of enums corresponding with modules that initiates the call.*
- enum AlertNotificationType

  *Notification type of alert,.*
- enum AlertRecurringType

  *Represents different recurring type of alert i.e., Yearly, Monthly, Weekly or Daily.*
- enum AlertRunType

  *Represents the type of alert depending upon the run type whether to run once or to run in a fixed recurring manner.*
- enum AlertStatus

  *Represents the status of an alert depending upon its state in its life cycle.*

### 4.7.1 Detailed Description

This package contains enums needed for Alert module.

## 4.8 Package alert.service

This package contains service layer classes needed for Alert Module.

### Classes

- class AlertExecuteStrategyFactory

  *This factory is going to pick the best available strategy to execute the alert so that the user can be notified.*
- class AlertExecutionScheduler

  *This is a background service that runs all the time.*
- class AlertService

  *Alert Service is a base class that is extended by other module specific services.*
- class AlertServiceFactory

  *AlertServiceFactory initializes appropriate alert service depending upon the parameter being passed.*
- interface IAlertExecuteStrategy

  *An interface for executing different type of alert based on alert type.*
- class RouteAlertExecuteStrategy

  *A concrete implementation of IAlertExecuteStrategy that handles executing alert related to route changes.*
- class ScheduleAlertExecuteStrategy

  *A concrete implementation of IAlertExecuteStrategy that handles executing alert related to user's regular schedule.*
- class TrackingAlertService

*Extends AlertService and provides CRUD methods to operate on alerts as well as sends alerts to execute via appropriate strategy.*

- class TransitAlertService

  *Extends AlertService and provides CRUD methods to operate on alerts as well as sends alerts to execute via appropriate strategy.*

- class UserAlertService

  *Extends AlertService and provides CRUD methods to operate on alerts as well as sends alerts to execute via appropriate strategy.*

### 4.8.1 Detailed Description

This package contains service layer classes needed for Alert Module. Service layer handles the workflow and orchestrates the calls to process the request. Depending upon the modules initiating the call, service implementation can invoke factory method in domain layer to create alert and can manipulate its properties. Also service layer can execute the alert using different strategy. Service layer uses factory and strategy patterns to select appropriate service implementation and strategy type.

## 4.9 Package common

This package contains common BusBuddy objects and utilities to be used by all modules.

### Classes

- class BaseController

  *This is a base class to be extended by each of the controller classes.*

- class BusBuddyBadRequestException

  *This exception object represents internal errors which may occur as a result of an error in the client's request.*

- class BusBuddyConflictException

  *This exception object is thrown when a request would create a conflict which violates constraints set within the system.*

- class BusBuddyException

  *This exception object is an abstract base class.*

- class BusBuddyForbiddenException

  *This exception object represents internal errors which may occur as a result of attempts to access a resource without authorization.*

- class BusBuddyInternalException

  *This exception object represents internal errors which may occur, which are generally not due to the specifics of what appears to be a valid request.*

- class BusBuddyNotFoundException

  *This exception object represents the error that occurs when a resource cannot be found.*

- class HashUtility

  *This is a utility class to handle secure hashes.*

- class MessageDeliveryUtility

  *This is a utility class to handle message delivery, such as through email or SMS.*

- interface Specification< T >

  *A Generic Specification to be used for chaining business validation rules together.*

### 4.9.1 Detailed Description

This package contains common BusBuddy objects and utilities to be used by all modules.

## 4.10 Package tracking

Tracking Module manages GPS vehicle tracking and alerting subscribed users based on vehicle location.

### Classes

- class AlertRangeLogic

  *Alert Range Logic implements the business logic to determine if a vehicle is within a range where an alert needs to be sent to a user who has registered for tracking alerts.*
- interface AlertSpecification

  *Interface for Alert Specifications which contain the business logic used to determine if an alert should be triggered for a vehicle.*
- enum AlertType

  *Enumeration of the alert types recognized by bus buddy.*
- class BusVehicle

  *Bus Vehicle is a concrete implementation of the abstract Transit Vehicle.*
- class CommercialTracking

  *Implements Subject GPSLocationTracking for retrieving GPS location updates from outside commercial tracking services.*
- class DelayAlertLogic

  *Implements the business logic to determine if the vehicle is behind schedule or not reporting GPS updates and send a notification to the transit company.*
- class GPSLocationObject

  *GPS Location is a value object used for GPS coordinates and the time of the last update.*
- class GPSLocationObserver

  *Observer Pattern - Observer interface for GPS location tracking.*
- class GPSLocationTracking

  *GPSLocationTracking - interface Subject of the Observer Pattern Defines methods for an observer GPS Device to register and receive updates on vehicle location.*
- class GPSPuller

  *GPS Puller is a concrete implementation of GPSLocationTracking for obtaining coordinates directly from a GPS device installed in a registered vehicle.*
- class GPSPusher

  *Implements Subject GPSLocationTracking for retrieving GPS location updates from registered vehicles.*
- class GPSVehicleTracker

  *Implementation of the Observer, update transit vehicle GPS location and time GPS Vehicle Tracker gets the state as new GPS coordinates and time from GPS Location Tracking and updates the transit vehicle.*
- interface ITrackingService

  *Interface for the Tracking Service Controller.*
- class TrackingAlertFactory

  *The Alert Factory handles the creation of a user alert.*
- class TrackingAlertObserver

  *Abstract class defining the methods for the tracking alert observer.*
- class TrackingDelayAlert

  *Tracking Alert Observer implements the abstract tracking alert observer and provides the method to actually send an alert to a registered user that their bus is approaching their stop.*
- class TrackingLocationAlert

  *Concrete implementation of the tracking alert observer.*
- class TrackingResponseModel

  *This is a basic tracking response model that is returned for every tracking related request.*
- class TrackingServiceController

  *Tracking service controller is the concrete implementation of the tracking service interface.*

- class TransitVehicle

  *Abstract transit vehicle class contains the common data for all types of vehicles and the Subject GPS Tracking and the GPS observer to receive GPS location updates.*
- class TransitVehicleFactory

  *Transit Vehicle Factory encapsulates the complexity of creating a new vehicle.*
- class UserTrackingAlertObject

  *User tracking alert information obtained from the user interface when the user registers for an alert.*
- class VehicleObject

  *Value Object containing vehicle information obtained when the user registers a vehicle using the user interface.*
- class VehicleRepository

  *Repository for information on vehicles registered on a route.*

### 4.10.1 Detailed Description

Tracking Module manages GPS vehicle tracking and alerting subscribed users based on vehicle location. The Tracking Module handles the tracking GPS data from the transit vehicles and initiating alerts to users subscribed to route based messages and triggers transit vehicle delay alerts. The transit vehicle location data is also available to the Maps Module to place the bus icon at the correct position on maps.

## 4.11 Package transit

The Transit Module is an interface to get Route/Fare/Detour information from a TransitProvider.

### Classes

- class AbstractFeedParserTemplate

  *A Template Method pattern to allow for the import of data from different TransitProviders in potentially different formats.*
- class Detour

  *A disruption in service due to an unexpected event.*
- class Fare

  *An immutable Value Object representing the cost, or 'fare,' required to ride a TransitVehicle on a particular Route.*
- class GoogleTransitServiceAdapter

  *An Adapter Class to allow a GoogleTransitServiceAPI service to appear as a TransitService.*
- interface GoogleTransitServiceAPI

  *A client to Google's* `Maps API`.
- class GTFSFeedParser

  *A AbstractFeedParserTemplate implementation designed to parse* `GTFS` *format ZIP files into Routes.*
- class InvalidRouteParseException

  *An InvalidRouteParseException indicates an invalid batch of parsed Routes has been been detected.*
- class ITeamTransitServiceController

  *The iTeam implementation of the TransitService that exposes Transit data via a REST Service.*
- class ITeamTripServiceController

  *An iTeam implementation of the TripService that exposes Trip data via a REST Service.*
- class Location

  *An immutable Value Object representing a physical point on the geographic coordinate system.*
- class PersistedTransitFeed

  *An implementation of the TransitFeed interface that communicates with a RouteRepository to retrieve its data.*
- class Route

  *A Route is a TransitVehicle path of travel, or a "Line," as referred to by a TransitProvider.*
- class RouteDisruptionAlert

*An Alert indicating a disruption of normal Route availability or scheduling.*

- class RouteDisruptionEvent

  *An event indicating a disruption in normal Route scheduling or service.*
- interface RouteRepository

  *A Repository Pattern supporting lifecycle operations of Routes, such as Read, Save, Delete, and Query functionality.*
- class RouteSpecification

  *A Specification Pattern class for validating a Route.*
- class Stop

  *A point on a Route in which a TransitVehicle will stop to pick up and drop off passengers.*
- interface TransitFeed

  *A TransitFeed is an abstraction over a service or set of services that provide information about Routes.*
- class TransitInfo

  *An immutable Value Object describing metadata about a TransitService.*
- class TransitProvider

  *A TransitProvider is a description of a company or organization that is the producer of public transportation services.*
- interface TransitProviderObserver

  *An asynchronous update interface for receiving notifications about TransitProvider Route disruptions.*
- interface TransitService

  *The TransitService is an interface to get Route/Fare/Detour information from a TransitProvider.*
- class Trip

  *A Trip is considered an ordered collection of Routes going from a starting point to an ending point.*
- interface TripService

  *A Service to calculate a collection of Routes, or a Trip, allowing for a continuous transit path from a start Location to an end Location.*

### 4.11.1 Detailed Description

The Transit Module is an interface to get Route/Fare/Detour information from a TransitProvider. The main module interface, the TransitService, provides a consistent interface for the application logic to query for this information.

From a design perspective, there are two main tasks performed by the Transit Module: Consuming Transit Information from a TransitProvider, and Providing Transit Information to Users.

## 4.12 Package user

This package contains the objects used by the User Module of the BusBuddy application.

### Classes

- class FavoriteTransitService

  *This class is a single transit service in a user's list of favorites.*
- class ITeamUserFavoritesService

  *This is the iTeam's implementation of UserFavoritesService.*
- class ITeamUserLoginService

  *This is the iTeam's implementation of UserLoginService.*
- class ITeamUserManagementService

  *This is the iTeam's implementation of UserManagementService.*
- class Session

  *This class represents a single session for a user of the system, and all of the state data associated with that session.*
- class SessionRepository

*This class is responsible for handling database access for Sessions, and to construct, persist, and retrieve Session objects.*

- class User

  *This class represents a single user of the system, and all of the state data associated with that user.*

- class UserFavoritesList

  *This class ties a userId to that user's ordered list of favorites.*

- class UserFavoritesRepository

  *This class is responsible for handling database access for favorites, and to persist and retrieve UserFavoritesList objects.*

- interface UserFavoritesService

  *This is the generic BusBuddy UserFavoritesService interface.*

- interface UserLoginService

  *This is the generic BusBuddy UserLoginService interface.*

- interface UserManagementService

  *This is the generic BusBuddy UserManagementService interface.*

- class UserRepository

  *This class is responsible for handling database access for User objects, and to construct, persist, and retrieve User objects.*

- enum UserType

  *This is an enumeration of the different statuses that a user can be assigned.*

### 4.12.1 Detailed Description

This package contains the objects used by the User Module of the BusBuddy application.

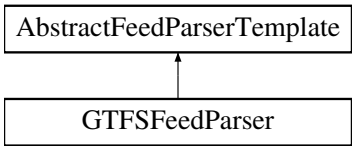
# Chapter 5

# Class Documentation

## 5.1 AbstractFeedParserTemplate Class Reference

A Template Method pattern to allow for the import of data from different TransitProviders in potentially different formats.

Inheritance diagram for AbstractFeedParserTemplate:




### Public Member Functions

- RouteRepository **getRouteRepository** ()
- void **setRouteRepository** (RouteRepository routeRepository)
- Specification< Route > **getRouteSpecification** ()
- void **setRouteSpecification** (Specification< Route > routeSpecification)

### Protected Member Functions

- void start (URL location) throws InvalidRouteParseException

  *The start method initiates the process and calls the appropriate methods in the appropriate order.*
- InputStream loadFeed (URL location)

  *Converts the resource URL into an InputStream for further processing.*
- abstract Set< Route > parseFeed (InputStream feed)

  *Parses the feed InputStream into a Set of Routes.*
- boolean validate (Route route)

  *Allow subclasses to validate Routes as they are parsed.*
- void saveRoutes (Set< Route > routes)

  *Save the Routes to the RouteRepository.*

### Private Attributes

- RouteRepository routeRepository

  *The RouteRepository dependency allows for the persistence of the parsed Routes.*

- Specification< Route > routeSpecification

  *This Specification allows subclasses to validate Routes as they are parsed.*

### 5.1.1 Detailed Description

A Template Method pattern to allow for the import of data from different TransitProviders in potentially different formats.

The algorithm sequence is as follows:

1. A URL of a resource location is passed into the start method. This method initiates the parsing/transformation process.
2. The start method calls the method loadFeed to establish the InputStream.
3. The InputStream returned by loadFeed is passed into the abstract parseFeed method. Subclasses will implement this as necessary to produce the resulting Routes
4. The newly created Routes are saved to the RouteRepository via the saveRoutes method.

### 5.1.2 Member Function Documentation

#### 5.1.2.1 InputStream loadFeed ( URL *location* ) [protected]

Converts the resource URL into an InputStream for further processing.

**Precondition**

**location** exists and has been validated.

**Parameters**

| | |
|---|---|
| *location* | The resource location |

**Returns**

The resulting InputStream

#### 5.1.2.2 abstract Set<Route> parseFeed ( InputStream *feed* ) [protected], [pure virtual]

Parses the feed InputStream into a Set of Routes.

Subclasses will implement this abstract method with the appropriate parsing logic for the particular input format.

**Parameters**

| | |
|---|---|
| *feed* | The resource InputStream |

**Returns**

The resulting Set of Routes

Implemented in GTFSFeedParser.

#### 5.1.2.3 void saveRoutes ( Set< Route > *routes* ) [protected]

Save the Routes to the RouteRepository.

**Precondition**

**routes** may be an empty Set, but must not be null.

**Parameters**

| | |
|---|---|
| *routes* | The Set of Routes to persist. |

#### 5.1.2.4 void start ( URL *location* ) throws InvalidRouteParseException [protected]

The start method initiates the process and calls the appropriate methods in the appropriate order.

**Exceptions**

| | |
|---|---|
| *InvalidRouteParseException* | if any of the parsed Routes fail to validate via the given routeSpecification. |

**Parameters**

| | |
|---|---|
| *location* | The input data resource location. This may be a local file or a remote resource. |

#### 5.1.2.5 boolean validate ( Route *route* ) [protected]

Allow subclasses to validate Routes as they are parsed.

Subclasses are encouraged to use this method

**Parameters**

| | |
|---|---|
| *route* | the route |

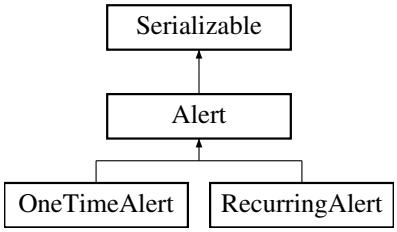
**Returns**

true, if successful

## 5.2 Alert Class Reference

This is a base Alert Model that has most of the common information about an Alert.

Inheritance diagram for Alert:



### Public Member Functions

- String **getAlertGuid** ()
- void **setAlertGuid** (String alertGuid)
- String **getDescription** ()
- void **setDescription** (String description)
- Date **getCreatedDateTime** ()
- void **setCreatedDateTime** (Date createdDateTime)
- Date **getStartDateTime** ()
- void **setStartDateTime** (Date startDateTime)
- Date **getExpireDateTime** ()
- void **setExpireDateTime** (Date expireDateTime)
- AlertStatus **getStatus** ()
- void **setStatus** (AlertStatus status)
- int **getErrorCount** ()
- void **setErrorCount** (int errorCount)
- AlertNotificationType **getAlertType** ()
- void **setAlertType** (AlertNotificationType alertType)
- AlertRunType **getAlertRunType** ()
- void **setAlertRunType** (AlertRunType alertRunType)
- UserInformation **getUserInformation** ()
- void **setUserInformation** (UserInformation userInformation)
- TripInformation **getTripInformation** ()
- void **setTripInformation** (TripInformation tripInformation)
- AlertInitiator **getAlertInitiator** ()
- void **setAlertInitiator** (AlertInitiator alertInitiator)

### Private Attributes

- String alertGuid

  *A unique identifier for Alert.*
- String description

  *A text description about the alert that the user or other modules want to remember.*
- Date createdDateTime

  *DateTime that the alert was created.*
- Date startDateTime

  *DateTime that the alert should start running.*
- Date expireDateTime

  *DateTime that the alert would expire.*
- AlertStatus Status

  *current status of the alert.*
- int errorCount

  *Count of error occurrence when the alert was ran.*
- AlertNotificationType alertType

  *Notification type of alert.*
- AlertRunType alertRunType

  *Run type of alert e.g., one time or recurring.*
- AlertInitiator alertInitiator

  *Module that initiated or created this alert.*
- UserInformation userInformation

  *Information about the user if the alert is created by a dedicated user.*
- TripInformation tripInformation

  *A HashMap of.*

### Static Private Attributes

- static final long **serialVersionUID** = -5671884600600864426L

### 5.2.1 Detailed Description

This is a base Alert Model that has most of the common information about an Alert.

OneTimeAlert and RecurringAlert extends this Alert Model. Alert can only be created from AlertFactory and then manipulated from AlertRepository.

### 5.2.2 Member Data Documentation

#### 5.2.2.1 AlertInitiator alertInitiator [private]

Module that initiated or created this alert.

AlertInitiator

#### 5.2.2.2 AlertRunType alertRunType [private]

Run type of alert e.g., one time or recurring.

Value is defined by AlertRunType

#### 5.2.2.3 AlertNotificationType alertType [private]

Notification type of alert.

Depends upon the value as specified in AlertNotificationType

#### 5.2.2.4 AlertStatus Status [private]

current status of the alert.

The value depends upon AlertStatus enum.

#### 5.2.2.5 UserInformation userInformation [private]

Information about the user if the alert is created by a dedicated user.

For any alerts generated by Transit or Tracking module, the user can either be admin of that module or the admin of the bus service provider. UserInformation

## 5.3 AlertExecuteStrategyFactory Class Reference

This factory is going to pick the best available strategy to execute the alert so that the user can be notified.

### Static Public Member Functions

- static IAlertExecuteStrategy getAlertExecuteStrategy (Alert alertModel)

  *Based on the alertModel information, this method is going to pick the best strategy to send the alert.*

### 5.3.1 Detailed Description

This factory is going to pick the best available strategy to execute the alert so that the user can be notified.

### 5.3.2 Member Function Documentation

#### 5.3.2.1 static IAlertExecuteStrategy getAlertExecuteStrategy ( Alert *alertModel* ) [static]

Based on the alertModel information, this method is going to pick the best strategy to send the alert.

**Parameters**

| | |
|---|---|
| *alertModel* | Alert model which is used to pick the best strategy. |

**Returns**

An implementation of IAlertExecuteStrategy that can send the notification message for given alert.

## 5.4 AlertExecutionScheduler Class Reference

This is a background service that runs all the time.

### Public Member Functions

- void runAlert ()

  *This method runs in fixed time interval indefinitely as long as program is alive.*

### Private Member Functions

- List< AlertService > getAlertServices ()

  *Gets all the classes that extends the AlertService via reflection.*

### 5.4.1 Detailed Description

This is a background service that runs all the time.

The runAlert() method is invoked in a fixed time interval which actually calls other methods and services to process the alert.

### 5.4.2 Member Function Documentation

#### 5.4.2.1 List<AlertService> getAlertServices ( ) [private]

Gets all the classes that extends the AlertService via reflection.

**Returns**

A list of classes extending AlertService.

#### 5.4.2.2 void runAlert ( )

This method runs in fixed time interval indefinitely as long as program is alive.

It calls getAlertServices() to get a list of classes that extend AlertService. It then loops through those services to call AlertService#sendAlert() to push notification to users preferred mode of communication.

## 5.5 AlertFactory Class Reference

An alert factory that can create different types of semi-populated alert models depending upon the information.

### Public Member Functions

- Alert createAlert (AlertRunType runType)

  *Creates an alert model depending upon the run Type.*
- Alert createAlert (AlertRecurringType recurringType)

  *Created alert model depending upon the recurringType.*

### 5.5.1 Detailed Description

An alert factory that can create different types of semi-populated alert models depending upon the information.

Any additional create method can be created during implementation phase.

### 5.5.2 Member Function Documentation

#### 5.5.2.1 Alert createAlert ( AlertRunType *runType* )

Creates an alert model depending upon the run Type.

**Parameters**

| | |
|---|---|
| *runType* | AlertRunType enum. |

**Returns**

Either a Onetime or Recurring alert Model.

#### 5.5.2.2 Alert createAlert ( AlertRecurringType *recurringType* )

Created alert model depending upon the recurringType.

**Parameters**

| | |
|---|---|
| *recurringType* | AlertRecurringType enum |

**Returns**

A recurring alert Model

## 5.6 AlertInitiator Enum Reference

A list of enums corresponding with modules that initiates the call.

### Public Attributes

- UserModule
  *User module initiates the call.*
- TrackingModule
  *Tracking module initiates the call.*
- TransitModule
  *Transit module initiates the call.*

### 5.6.1 Detailed Description

A list of enums corresponding with modules that initiates the call.

## 5.7 AlertNotificationType Enum Reference

Notification type of alert,.

### Public Attributes

- PlannedDisruption
  *A planned alert that is to be run in the future.*
- UnplannedDisruption
  *A sudden change in route or plan can trigger this alert.*
- ScheduleInformation
  *A general type of alert where a user is to be notified of their schedule in a timely fashion.*

### 5.7.1 Detailed Description

Notification type of alert,.

### 5.7.2 Member Data Documentation

#### 5.7.2.1 PlannedDisruption

A planned alert that is to be run in the future.

Usually this is provided by Transit Module when the bus route is going to be changed in near future.

#### 5.7.2.2 ScheduleInformation

A general type of alert where a user is to be notified of their schedule in a timely fashion.

Generally, user module creates this type of alert.

#### 5.7.2.3 UnplannedDisruption

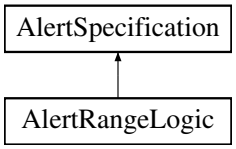
A sudden change in route or plan can trigger this alert.

Usually this type of alert is executed right away. This type of alert is provided by either Transit or Tracking module.

## 5.8 AlertRangeLogic Class Reference

Alert Range Logic implements the business logic to determine if a vehicle is within a range where an alert needs to be sent to a user who has registered for tracking alerts.

Inheritance diagram for AlertRangeLogic:




### Public Member Functions

- boolean inAlertRange (GPSLocationObject vehicleLocation)

  *Provides the logic necessary to determine from the GPS coordinates if the registered user should be alerted.*

### 5.8.1 Detailed Description

Alert Range Logic implements the business logic to determine if a vehicle is within a range where an alert needs to be sent to a user who has registered for tracking alerts.

This logic is designed to guarantee that an alert will be sent to the user before the vehicle has passes the desired stop. GPS coordinates are regularly updated, but not necessarily in real-time. BusBuddy needs to notify the user before the bus reaches the stop. An alert range is a distance range before the stop when the user should be notified.

## 5.9 AlertRecurringType Enum Reference

Represents different recurring type of alert i.e., Yearly, Monthly, Weekly or Daily.

### Public Attributes

- **Yearly**
- **Monthly**
- **Weekly**
- **Daily**

### 5.9.1 Detailed Description

Represents different recurring type of alert i.e., Yearly, Monthly, Weekly or Daily.

## 5.10 AlertRepository Class Reference

A Repository that handles the persistent behavior of the Alert aggregate.

### Public Member Functions

- Alert saveAlert (Alert alertModel)

  *This methods take an Alert and saves it to the database.*
- boolean deleteAlert (Alert alertModel)

  *This method deletes the alert that is being passed.*
- Alert updateAlert (Alert alertModel)

  *This method is used to update the alert with new information.*
- List< Alert > getAlertByDateTime (Date dateTimeToFetch, int offsetMinute)

  *This method fetches all the alerts that is to be run in next couple of minutes of given date and time.*
- List< Alert > getAlertByRoute (String routeId)

  *This method fetches all the alerts that is linked to the routeId.*
- List< Alert > getAlertByUserId (String userId)

  *This method fetches all the alerts that is tied to a user.*

### 5.10.1 Detailed Description

A Repository that handles the persistent behavior of the Alert aggregate.

It has methods that can alter the lifecycle of the aggregate.

### 5.10.2 Member Function Documentation

#### 5.10.2.1 boolean deleteAlert ( Alert *alertModel* )

This method deletes the alert that is being passed.

**Precondition**

the alertModel being passed at least needs to have an ID defined.

**Postcondition**

the alert will be removed from the system and can no longer be accessed.

**Parameters**

| | |
|---|---|
| *alertModel* | The Alert model that is to be deleted. |

**Returns**

A boolean to indicate whether the delete was success or not.

#### 5.10.2.2 List<Alert> getAlertByDateTime ( Date *dateTimeToFetch*, int *offsetMinute* )

This method fetches all the alerts that is to be run in next couple of minutes of given date and time.

e.g., if DateTime is NOW and offset is 5 minutes. Then it fetches all the alerts that is to be run in next 5 minutes.

**Parameters**

| | |
|---|---|
| *dateTimeToFetch* | DateTime when the alert is supposed to run. |
| *offsetMinute* | An int value that is used to fetch alerts within that minute in future. |

**Returns**

Returns a list of Alert models that is to be run in next couple of minutes (offsetMinute) of given date time.

#### 5.10.2.3 List<Alert> getAlertByRoute ( String *routeId* )

This method fetches all the alerts that is linked to the routeId.

**Parameters**

| | |
|---|---|
| *routeId* | The route ID that is being affected. |

**Returns**

A list of Alert models.

#### 5.10.2.4 List<Alert> getAlertByUserId ( String *userId* )

This method fetches all the alerts that is tied to a user.

**Parameters**

| | |
|---|---|
| *userId* | userId that is being affected |

**Returns**

A list of Alert models.

#### 5.10.2.5 Alert saveAlert ( Alert *alertModel* )

This methods take an Alert and saves it to the database.

**Parameters**

| | |
|---|---|
| *alertModel* | Alert model to be saved |

**Returns**

The saved object with updated property.

Save the alert via Hibernate.

#### 5.10.2.6 Alert updateAlert ( Alert *alertModel* )

This method is used to update the alert with new information.

**Precondition**

the alert must exist in the system.

**Parameters**

| | |
|---|---|
| *alertModel* | A Alert model that needs to be updated |

**Returns**

Returns the updated Alert model back to the method that is calling.

## 5.11 AlertRequestController Class Reference

This is a front facing controller that takes in request from other module via REST call and returns a JSON representation data.

### Public Member Functions

- AlertResponseModel processUserAlertRequest (UserAlertRequestModel userAlertRequest, String encryptedToken)

  *Processes Alert manipulation request from User module.*
- AlertResponseModel processTransitAlertRequest (TransitAlertRequestModel transitAlertRequest, String encryptedToken)

  *Processes Alert manipulation request from Transit module.*
- AlertResponseModel processTrackingAlertRequest (TrackingAlertRequestModel trackingAlertRequest, String encryptedToken)

  *Processes Alert manipulation request from Tracking module.*

### Private Member Functions

- void verifySession (AlertRequestModel requestModel, String encryptedToken)

  *Method to verify the validity of session token being passed.*

### Private Attributes

- SessionVerificationFactory sessionVerificationFactory

  *Session Verification Factory picks the appropriate type of session verification strategy depending upon which module is initiating the call.*

### 5.11.1 Detailed Description

This is a front facing controller that takes in request from other module via REST call and returns a JSON representation data.

User module calls the alert module to create new alert, delete alert, update alert or to get a list of alert for a user. Transit module can call alert module to create, delete and update a new alert. Tracking module can call alert module to create a new alert. As far as public facing end points go, we have implemented one endpoint for each module. During implementation there might be different GET, POST, PUT and DELETE methods. But for now, we are putting only one method per module as a placeholder assuming that it can take different actions depending upon the information in the AlertRequestModel.

### 5.11.2 Member Function Documentation

#### 5.11.2.1 AlertResponseModel processTrackingAlertRequest ( TrackingAlertRequestModel *trackingAlertRequest,* String *encryptedToken* )

Processes Alert manipulation request from Tracking module.

**Parameters**

| | |
|---|---|
| *trackingAlert-Request* | { |

**See Also**

TrackingAlertRequestModel}

**Parameters**

| | |
|---|---|
| *encryptedToken* | An encrypted token that can be validated. |

**Returns**

An AlertResponseModel that has the success/error information.

#### 5.11.2.2 AlertResponseModel processTransitAlertRequest ( TransitAlertRequestModel *transitAlertRequest,* String *encryptedToken* )

Processes Alert manipulation request from Transit module.

**Parameters**

| | |
|---|---|
| *transitAlert-Request* | { |

**See Also**

TransitAlertRequestModel}

**Parameters**

| | |
|---|---|
| *encryptedToken* | • An encrypted token that can be validated. |

**Returns**

An AlertResponseModel that has the success/error information.

#### 5.11.2.3 AlertResponseModel processUserAlertRequest ( UserAlertRequestModel *userAlertRequest,* String *encryptedToken* )

Processes Alert manipulation request from User module.

**Parameters**

| | |
|---|---|
| *userAlert-Request* | An alertRequestModel that has the necessary information regarding creation of an alert. { |

**See Also**

UserAlertRequestModel}

**Parameters**

| | |
|---|---|
| *encryptedToken* | An encrypted token that can be validated. |

**Returns**

An AlertResponseModel that has the success/error information.

#### 5.11.2.4 void verifySession ( AlertRequestModel *requestModel,* String *encryptedToken* ) [private]

Method to verify the validity of session token being passed.

If token verification fails, it automatically throws an error the request is terminated.

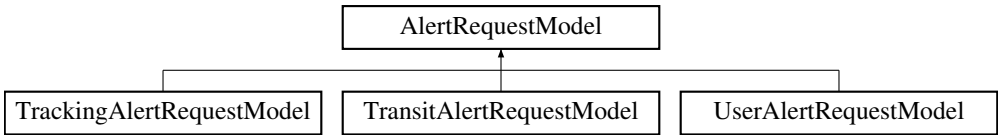
**Parameters**

| | |
|---|---|
| *requestModel* | An object representation of JSON that has the request. |
| *encryptedToken* | An encrypted token that can be validated. |

## 5.12 AlertRequestModel Class Reference

A base model that contains bare minimum information about the alert request.

Inheritance diagram for AlertRequestModel:




### Public Member Functions

- AlertInitiator **getAlertInitiator** ()
- void **setAlertInitiator** (AlertInitiator alertInitiator)

### Private Attributes

- AlertInitiator alertInitiator

  *Alert Initiator {.*

### 5.12.1 Detailed Description

A base model that contains bare minimum information about the alert request.

Any model that extends this class can add additional data that is needed.

### 5.12.2 Member Data Documentation

#### 5.12.2.1 AlertInitiator alertInitiator [private]

Alert Initiator {.

**See Also**

AlertInitiator}.

## 5.13 AlertResponseModel Class Reference

This is a basic alert response model that is returned for every alert related requested.

### Private Attributes

- String status

  *Status message for the alert process job.*
- String errorMessage

  *Any error message if the alert request fails.*

### 5.13.1 Detailed Description

This is a basic alert response model that is returned for every alert related requested.

Additional fields can be added as needed during implementation phase.

## 5.14 AlertRunType Enum Reference

Represents the type of alert depending upon the run type whether to run once or to run in a fixed recurring manner.

### Public Attributes

- **Onetime**
- **Recurring**

### 5.14.1 Detailed Description

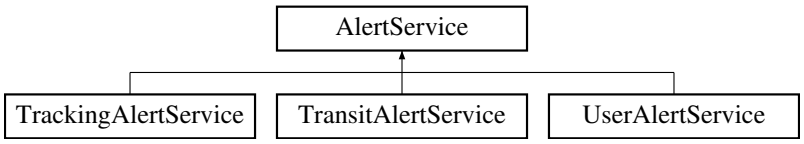
Represents the type of alert depending upon the run type whether to run once or to run in a fixed recurring manner.

OneTime Alert runs only once and is then deactivated. Recurring Alert runs in every certain amount of time depending upon the parameter defined.

## 5.15 AlertService Class Reference

Alert Service is a base class that is extended by other module specific services.

Inheritance diagram for AlertService:




### Public Member Functions

- AlertResponseModel saveAlert (Alert alertModel)

  *Takes in an Alert model to persist it into db.*
- AlertResponseModel deleteAlert (Alert alertModel)

  *Takes in an Alert Model for deletion.*
- abstract boolean sendAlert ()

  *Finds all the alerts that are initiated by this service and calls AlertExecuteStrategyFactory to find appropriate strategy to send each type of alert.*

### Protected Member Functions

- abstract AlertResponseModel createAlert (AlertRequestModel requestModel)

  *An abstract class that must be implemented by derived classes to create a new alert model.*
- abstract AlertResponseModel updateAlert (Alert alertModel)

  *Takes in a final Alert Model that needs to be updated in db.*

### Protected Attributes

- AlertExecuteStrategyFactory alertExecuteStrategyFactory

  *{*
- AlertRepository alertRepository

  *{*
- AlertFactory alertFactory

  *{*

## 5.15.1 Detailed Description

Alert Service is a base class that is extended by other module specific services.

This abstract class defines signature for basic CRUD operation.

## 5.15.2 Member Function Documentation

### 5.15.2.1 abstract AlertResponseModel createAlert ( AlertRequestModel *requestModel* ) [protected], [pure virtual]

An abstract class that must be implemented by derived classes to create a new alert model.

#### Precondition

An alertRequest model must be supplied that must have necessary information to create an alert

#### Parameters

| | |
|---|---|
| *requestModel* | A AlertResponseModel that has necessary information to create a new alert model. |

**Returns**

Returns an AlertReponseModel that contains information needed by the caller.

Implemented in TrackingAlertService, TransitAlertService, and UserAlertService.

#### 5.15.2.2 AlertResponseModel deleteAlert ( Alert *alertModel* )

Takes in an Alert Model for deletion.

**Parameters**

| | |
|---|---|
| *alertModel* | A valid Alert model |

**Returns**

An AlertResponseModel that has the success or error message.

#### 5.15.2.3 AlertResponseModel saveAlert ( Alert *alertModel* )

Takes in an Alert model to persist it into db.

**Parameters**

| | |
|---|---|
| *alertModel* | A valid Alert model |

**Returns**

An AlertResponseModel that has the success or error message.

#### 5.15.2.4 abstract boolean sendAlert ( ) [pure virtual]

Finds all the alerts that are initiated by this service and calls AlertExecuteStrategyFactory to find appropriate strategy to send each type of alert.

1. Get all the alerts created by this service/module. 2. Loop through each one of them and call appropriate strategy via StrategyFactory 3. Use execute method in strategy to push notification

**Returns**

Implemented in TrackingAlertService, TransitAlertService, and UserAlertService.

#### 5.15.2.5 abstract AlertResponseModel updateAlert ( Alert *alertModel* ) [protected], [pure virtual]

Takes in a final Alert Model that needs to be updated in db.

**Parameters**

| | |
|---|---|
| *alertModel* | A valid Alert model |

**Returns**

An AlertResponseModel that has the success or error message.

Implemented in TrackingAlertService, TransitAlertService, and UserAlertService.

### 5.15.3 Member Data Documentation

#### 5.15.3.1 AlertExecuteStrategyFactory alertExecuteStrategyFactory [protected]

{

**See Also**

AlertExecuteStrategyFactory}. This is autowired via Spring Framework.

#### 5.15.3.2 AlertFactory alertFactory [protected]

{

**See Also**

AlertFactory}. This is autowired via Spring framework.

#### 5.15.3.3 AlertRepository alertRepository [protected]

{

**See Also**

AlertRepository}. This is autowired via Spring Framework.

## 5.16 AlertServiceFactory Class Reference

AlertServiceFactory initializes appropriate alert service depending upon the parameter being passed.

### Public Member Functions

- AlertService getAlertService (AlertRequestModel requestModel)

  *This method returns a concrete implementation of the alert service.*

### Private Attributes

- UserAlertService userAlertService

  *{*
- TrackingAlertService trackingAlertService

  *{*
- TransitAlertService transitAlertService

  *{*

### 5.16.1 Detailed Description

AlertServiceFactory initializes appropriate alert service depending upon the parameter being passed.

Currently, each module is aligned to each service but in the future it could change such that 2 module can use same service.

### 5.16.2 Member Function Documentation

#### 5.16.2.1 AlertService getAlertService ( AlertRequestModel *requestModel* )

This method returns a concrete implementation of the alert service.

**Parameters**

| | |
|---|---|
| *requestModel* | A JSON representation of the request from controller. |

**Returns**

An extension of AlertService.

thrown an error if applicable.

### 5.16.3 Member Data Documentation

#### 5.16.3.1 TrackingAlertService trackingAlertService [private]

{

**See Also**

TrackingAlertService}. This is autowired via Spring Framework.

#### 5.16.3.2 TransitAlertService transitAlertService [private]

{

**See Also**

TransitAlertService}. This is autowired via Spring Framework.

#### 5.16.3.3 UserAlertService userAlertService [private]

{

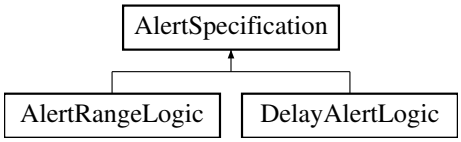
**See Also**

UserAlertService}. This is autowired via Spring Framework.

## 5.17 AlertSpecification Interface Reference

Interface for Alert Specifications which contain the business logic used to determine if an alert should be triggered for a vehicle.

Inheritance diagram for AlertSpecification:




### Public Member Functions

- boolean inAlertRange (GPSLocationObject vehicleLocation)

  *Compare GPS location and time with alert information to determine if vehicle is within range of a stop and user(s) need to be notified.*

### 5.17.1 Detailed Description

Interface for Alert Specifications which contain the business logic used to determine if an alert should be triggered for a vehicle.

Alert specifications are referenced in the vehicle tracking observer and used by the subject to determine when to send an alert. This is to reduce the number of false positive alerts.

### 5.17.2 Member Function Documentation

#### 5.17.2.1 boolean inAlertRange ( GPSLocationObject *vehicleLocation* )

Compare GPS location and time with alert information to determine if vehicle is within range of a stop and user(s) need to be notified.

**Parameters**

| | |
|---|---|
| *lastUpdateTime* | - Time GPS information was last updated |
| *vehicleLocation* | - Latest GPS coordinates obtained from a vehicle |

**Returns**

true if vehicle is in alert range, false if vehicle is not in alert range

Implemented in AlertRangeLogic, and DelayAlertLogic.

## 5.18 AlertStatus Enum Reference

Represents the status of an alert depending upon its state in its life cycle.

### Public Attributes

- Active

  *Represents an alert that is active and is ready to be run when time comes.*
- Deactive

  *Represents an alert that is in dormant state and wont run even if its time parameter is valid.*
- Running

  *Represents an alert that is in running state.*

- Expired

  *Represents an alert that is expired.*
- Error

  *Represents an alert that is in error state due to technical difficulties.*

### 5.18.1 Detailed Description

Represents the status of an alert depending upon its state in its life cycle.

### 5.18.2 Member Data Documentation

#### 5.18.2.1 Deactive

Represents an alert that is in dormant state and wont run even if its time parameter is valid.

Usually alert canbe in this status if it is paused.

#### 5.18.2.2 Error

Represents an alert that is in error state due to technical difficulties.

This alert will be run 3 times before it is permanently paused until further action from user or admin.

#### 5.18.2.3 Expired

Represents an alert that is expired.

This type of alert are deleted periodically.

## 5.19 AlertType Enum Reference

Enumeration of the alert types recognized by bus buddy.

### Public Attributes

- **LOCATION**
- **DELAY**

### 5.19.1 Detailed Description

Enumeration of the alert types recognized by bus buddy.

Bus Buddy can then use a configuration file to tie the alert type to the AlertSpecification to determine the logic necessary to determine if users registered to a vehicle should be sent alerts.

## 5.20 AlertUserClient Class Reference

Client layer will handle all the responsibility of sending request to other modules or external sources.

### Public Member Functions

- UserInformation getuserInformation (String userId, String sessionToken)

  *Calls User module to get User information for a particular user such as their contact preference via a REST call.*
- boolean verifySessionToken (String sessionToken)

  *Calls the user module to verify the authenticity of session token being passed.*

### Package Attributes

- String userModuleURL

  *A user module url that is used to make a REST call to get users information This information is saved in a config file that is retrieved via Spring's Resource annotation.*

### 5.20.1 Detailed Description

Client layer will handle all the responsibility of sending request to other modules or external sources.

This particular client class will handle request to User module to get necessary user information.

### 5.20.2 Member Function Documentation

#### 5.20.2.1 UserInformation getuserInformation ( String *userId,* String *sessionToken* )

Calls User module to get User information for a particular user such as their contact preference via a REST call.

**Parameters**

| | |
|---|---|
| *userId* | User ID of the user that should receive the alert. |
| *sessionToken* | Valid long lived session token that can be used to get information about a particular user. |

**Returns**

A UserInformation that contains user data.

gets the user information from the user module via REST call

#### 5.20.2.2 boolean verifySessionToken ( String *sessionToken* )

Calls the user module to verify the authenticity of session token being passed.

**Parameters**

| | |
|---|---|
| *sessionToken* | An encrypted token that needs to be validated. |

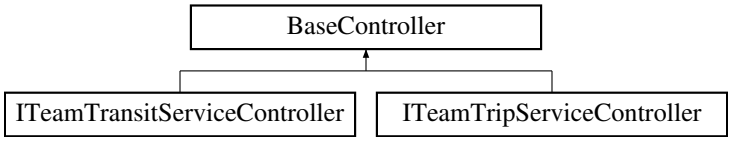
**Returns**

A boolean flag indicating the authenticity of the session token.

## 5.21 BaseController Class Reference

This is a base class to be extended by each of the controller classes.

Inheritance diagram for BaseController:




## Public Member Functions

- ResponseEntity< String > handleBusBuddyException (BusBuddyException e)

  *This method handles cases where BusBuddyException is thrown from controller methods.*
- ResponseEntity< String > handleGenericException (BusBuddyException e)

  *This method handles cases where a generic Exception is thrown from controller methods (other than BusBuddyException).*

### 5.21.1 Detailed Description

This is a base class to be extended by each of the controller classes.

This provides a means to handle exceptions that need to be thrown back up to the user. It could be modified to add other common logic that apply to multiple controllers.

### 5.21.2 Member Function Documentation

#### 5.21.2.1 ResponseEntity<String> handleBusBuddyException ( BusBuddyException *e* )

This method handles cases where BusBuddyException is thrown from controller methods.

It will format the exception for the user, and return the correct HTTP status code, based on the code stored within the exception.

**Parameters**

| | |
|---|---|
| *e* | exception which was thrown |

**Returns**

ResponseEntity object

#### 5.21.2.2 ResponseEntity<String> handleGenericException ( BusBuddyException *e* )

This method handles cases where a generic Exception is thrown from controller methods (other than BusBuddyException).

It will format the exception for the user, and return a generic HTTP 500. Since handled exceptions should result in a BusBuddyException, if this happens, it is unexpected behavior and should be treated as an internal error.

**Parameters**

| | |
|---|---|
| *e* | exception which was thrown |

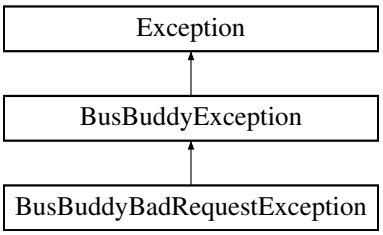
**Returns**

ResponseEntity object

## 5.22 BusBuddyBadRequestException Class Reference

This exception object represents internal errors which may occur as a result of an error in the client's request.

Inheritance diagram for BusBuddyBadRequestException:




### Public Member Functions

- **BusBuddyBadRequestException** (String message)
- **BusBuddyBadRequestException** (Throwable cause)
- **BusBuddyBadRequestException** (String message, Throwable cause)
- HttpStatus getHttpCode ()

  *This method returns the HTTP status code associated with this exception.*

### Static Private Attributes

- static final long **serialVersionUID** = -5974225882272455539L

### Additional Inherited Members

### 5.22.1 Detailed Description

This exception object represents internal errors which may occur as a result of an error in the client's request.

### 5.22.2 Member Function Documentation

#### 5.22.2.1 HttpStatus getHttpCode ( ) [virtual]

This method returns the HTTP status code associated with this exception.

**Returns**

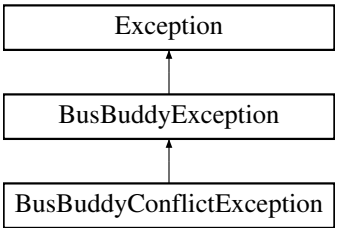
HTTP 400 Bad Request

Implements BusBuddyException.

## 5.23 BusBuddyConflictException Class Reference

This exception object is thrown when a request would create a conflict which violates constraints set within the system.

Inheritance diagram for BusBuddyConflictException:




### Public Member Functions

- **BusBuddyConflictException** (String message)
- **BusBuddyConflictException** (Throwable cause)
- **BusBuddyConflictException** (String message, Throwable cause)
- HttpStatus getHttpCode ()

  *This method returns the HTTP status code associated with this exception.*

### Static Private Attributes

- static final long **serialVersionUID** = -2044397352042431762L

### Additional Inherited Members

#### 5.23.1 Detailed Description

This exception object is thrown when a request would create a conflict which violates constraints set within the system.

#### 5.23.2 Member Function Documentation

##### 5.23.2.1 HttpStatus getHttpCode ( ) [virtual]

This method returns the HTTP status code associated with this exception.

**Returns**

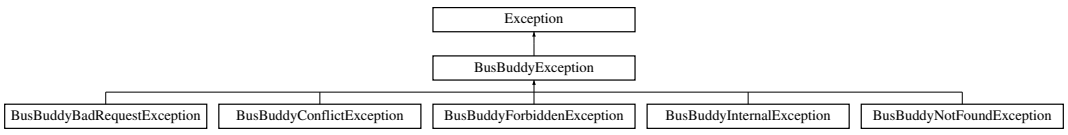
HTTP 409 Conflict

Implements BusBuddyException.

## 5.24 BusBuddyException Class Reference

This exception object is an abstract base class.

Inheritance diagram for BusBuddyException:



### Protected Member Functions

- **BusBuddyException** (String message)
- **BusBuddyException** (Throwable cause)
- **BusBuddyException** (String message, Throwable cause)
- abstract HttpStatus getHttpCode ()

  *This method returns a Spring HTTP status code object representing the HTTP status code tied to this exception.*

### Static Private Attributes

- static final long **serialVersionUID** = 5906063726935813830L

### 5.24.1 Detailed Description

This exception object is an abstract base class.

Other excpetions within the BusBuddy application will extend this class. This provides a common base for all application exceptions.

### 5.24.2 Member Function Documentation

#### 5.24.2.1 abstract HttpStatus getHttpCode ( ) [protected], [pure virtual]

This method returns a Spring HTTP status code object representing the HTTP status code tied to this exception.

**Returns**

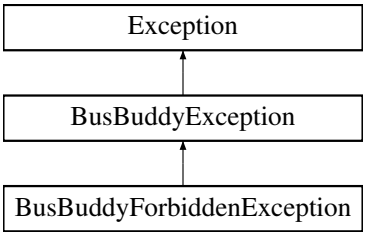
HTTP Status Code object

Implemented in BusBuddyConflictException, BusBuddyForbiddenException, BusBuddyInternalException, BusBuddyBadRequestException, and BusBuddyNotFoundException.

## 5.25 BusBuddyForbiddenException Class Reference

This exception object represents internal errors which may occur as a result of attempts to access a resource without authorization.

Inheritance diagram for BusBuddyForbiddenException:




### Public Member Functions

- **BusBuddyForbiddenException** (String message)
- **BusBuddyForbiddenException** (Throwable cause)
- **BusBuddyForbiddenException** (String message, Throwable cause)
- HttpStatus getHttpCode ()

  *This method returns the HTTP status code associated with this exception.*

### Static Private Attributes

- static final long **serialVersionUID** = -4463973248172436949L

### Additional Inherited Members

### 5.25.1 Detailed Description

This exception object represents internal errors which may occur as a result of attempts to access a resource without authorization.

### 5.25.2 Member Function Documentation

#### 5.25.2.1 HttpStatus getHttpCode ( ) [virtual]

This method returns the HTTP status code associated with this exception.

**Returns**

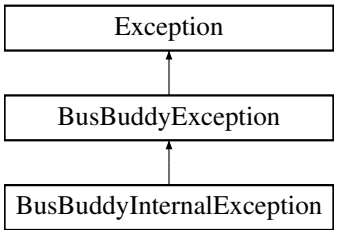
HTTP 403 Forbidden

Implements BusBuddyException.

## 5.26 BusBuddyInternalException Class Reference

This exception object represents internal errors which may occur, which are generally not due to the specifics of what appears to be a valid request.

Inheritance diagram for BusBuddyInternalException:




### Public Member Functions

- **BusBuddyInternalException** (String message)
- **BusBuddyInternalException** (Throwable cause)
- **BusBuddyInternalException** (String message, Throwable cause)
- HttpStatus getHttpCode ()

  *This method returns the HTTP status code associated with this exception.*

### Static Private Attributes

- static final long **serialVersionUID** = 4549592428602851924L

### Additional Inherited Members

#### 5.26.1 Detailed Description

This exception object represents internal errors which may occur, which are generally not due to the specifics of what appears to be a valid request.

#### 5.26.2 Member Function Documentation

##### 5.26.2.1 HttpStatus getHttpCode ( ) [virtual]

This method returns the HTTP status code associated with this exception.

**Returns**

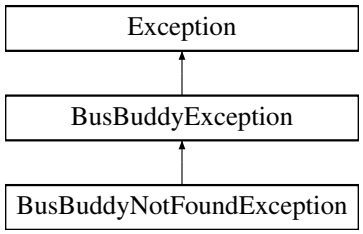
HTTP 500 Internal Server Error

Implements BusBuddyException.

## 5.27 BusBuddyNotFoundException Class Reference

This exception object represents the error that occurs when a resource cannot be found.

Inheritance diagram for BusBuddyNotFoundException:




### Public Member Functions

- **BusBuddyNotFoundException** (String message)
- **BusBuddyNotFoundException** (Throwable cause)
- **BusBuddyNotFoundException** (String message, Throwable cause)
- HttpStatus getHttpCode ()

  *This method returns the HTTP status code associated with this exception.*

### Static Private Attributes

- static final long **serialVersionUID** = -5490492502661128777L

### Additional Inherited Members

#### 5.27.1 Detailed Description

This exception object represents the error that occurs when a resource cannot be found.

### 5.27.2 Member Function Documentation

#### 5.27.2.1 HttpStatus getHttpCode ( ) [virtual]

This method returns the HTTP status code associated with this exception.

**Returns**

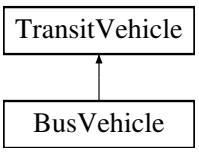
HTTP 404 Not Found

Implements BusBuddyException.

## 5.28 BusVehicle Class Reference

Bus Vehicle is a concrete implementation of the abstract Transit Vehicle.

Inheritance diagram for BusVehicle:




### Public Member Functions

- BusVehicle ()
  *Constructor for bus type vehicles, perform any initializations unique to buses.*
- void registerTrackingAlert (TrackingAlertObserver ao)
  *Register any user alerts for this vehicle.*
- void unregisterTrackingAlert (TrackingAlertObserver ao)
  *Unregister any user alert currently tracking this bus.*
- void checkForAlerts ()
  *When the bus GPS position is updated, determine if any user alerts need to be sent.*

### Private Attributes

- ArrayList< TrackingAlertObserver > alertList
  *List of alerts registered for this vehicle.*

### 5.28.1 Detailed Description

Bus Vehicle is a concrete implementation of the abstract Transit Vehicle.

Contains data and functionality specific to buses.

### 5.28.2 Member Data Documentation

#### 5.28.2.1 ArrayList<TrackingAlertObserver> alertList [private]

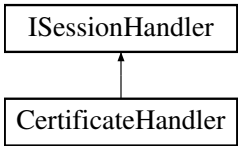
List of alerts registered for this vehicle.

Note alerts may be tracking or delay alerts

## 5.29 CertificateHandler Class Reference

A concrete strategy implementation of ISessionHandler that can verify the certificate token being passed.

Inheritance diagram for CertificateHandler:




### Public Member Functions

- boolean verifySessionToken (String sessionToken) throws BusBuddyForbiddenException

  *This method takes in an encrypted certificate from Tracking and Transit module in a form of string and then validates for its authenticity.*

### 5.29.1 Detailed Description

A concrete strategy implementation of ISessionHandler that can verify the certificate token being passed.

### 5.29.2 Member Function Documentation

#### 5.29.2.1 boolean verifySessionToken ( String *sessionToken* ) throws BusBuddyForbiddenException

This method takes in an encrypted certificate from Tracking and Transit module in a form of string and then validates for its authenticity.

**Exceptions**

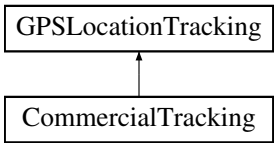
| | |
|---|---|
| *BusBuddyForbidden-Exception* | Throws an exception when the token is not valid. |

Implements ISessionHandler.

## 5.30 CommercialTracking Class Reference

Implements Subject GPSLocationTracking for retrieving GPS location updates from outside commercial tracking services.

Inheritance diagram for CommercialTracking:




### Classes

- class CommercialTrackingHolder

*Commercial Tracking Holder is loaded on the first execution of CommercialTracking.getInstance() or the first access to CommercialTracking.INSTANCE, not before (lazy instantiation).*

### Protected Member Functions

- void registerGPSDevice (GPSLocationObserver gpsObs)

  *Register a GPS Device to the list to poll for updates.*
- void unregisterGPSDevice (GPSLocationObserver gpsObs)

  *Remove a GPS device from the list currently being polled for updates.*
- void pollGPSDevice ()

  *Continuously poll the registered GPS devices for location updates.*

### Static Protected Member Functions

- static CommercialTracking getInstance ()

  *Instantiates a single Commercial Tracking service to the caller.*

### Private Member Functions

- CommercialTracking ()

  *Only need one Commercial Tracking Service running to track by polling all registered GPS devices.*

### Private Attributes

- ArrayList< GPSLocationObserver > gpsObserver

  *Array list of GPS devices registered for updates.*

## 5.30.1 Detailed Description

Implements Subject GPSLocationTracking for retrieving GPS location updates from outside commercial tracking services.

**Postcondition**

New GPS commercial tracker created or existing one returned.

## 5.30.2 Constructor & Destructor Documentation

### 5.30.2.1 CommercialTracking ( ) [private]

Only need one Commercial Tracking Service running to track by polling all registered GPS devices.

Constructor, creates ArrayList<GPSLocationObserver> to hold registered observers.

## 5.30.3 Member Function Documentation

### 5.30.3.1 static CommercialTracking getInstance ( ) [static], [protected]

Instantiates a single Commercial Tracking service to the caller.

**Returns**

- CommercialTracking instance

## 5.31 CommercialTracking.CommercialTrackingHolder Class Reference

Commercial Tracking Holder is loaded on the first execution of CommercialTracking.getInstance() or the first access to CommercialTracking.INSTANCE, not before (lazy instantiation).

### Static Public Attributes

- static final CommercialTracking **INSTANCE** = new CommercialTracking()

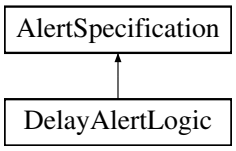
### 5.31.1 Detailed Description

Commercial Tracking Holder is loaded on the first execution of CommercialTracking.getInstance() or the first access to CommercialTracking.INSTANCE, not before (lazy instantiation).

## 5.32 DelayAlertLogic Class Reference

Implements the business logic to determine if the vehicle is behind schedule or not reporting GPS updates and send a notification to the transit company.

Inheritance diagram for DelayAlertLogic:




### Public Member Functions

- boolean inAlertRange (GPSLocationObject vehicleLocation)

  *Implements Subject GPSLocationTracking for retrieving GPS location updates from outside commercial tracking services.*

### 5.32.1 Detailed Description

Implements the business logic to determine if the vehicle is behind schedule or not reporting GPS updates and send a notification to the transit company.

## 5.33 Detour Class Reference

A disruption in service due to an unexpected event.

### Public Member Functions

- Detour (String detourId, String cause, int estimatedDelay, Set< Stop > affectedStops)

  *Instantiates a new Detour with all required fields.*
- String **getCause** ()
- void **setCause** (String cause)
- int **getEstimatedDelay** ()

- void **setEstimatedDelay** (int estimatedDelay)
- Set< Stop > **getAffectedStops** ()
- void **setAffectedStops** (Set< Stop > affectedStops)
- String **getDetourId** ()
- void **setDetourId** (String detourId)

### Private Attributes

- String detourId

  *A unique identifier for this Detour.*
- String cause

  *A text-based description of the cause of the Detour, intended to be displayed to customers.*
- int estimatedDelay

  *The estimated time (in minutes) that each of the Stops in the.*
- Set< Stop > affectedStops

  *All Stops that are subject to the noted estimatedDelay.*

## 5.33.1 Detailed Description

A disruption in service due to an unexpected event.

A Detour may not affect all Stops in a Route, as a Detour may only alter portions of the Route. Any affected Stop will be listed in the affectedStops attribute.

## 5.33.2 Constructor & Destructor Documentation

### 5.33.2.1 Detour ( String *detourId,* String *cause,* int *estimatedDelay,* Set< Stop > *affectedStops* )

Instantiates a new Detour with all required fields.

#### Precondition

**estimatedDelay** >= 0
**affectedStops**.size() > 0

#### Parameters

| | |
|---|---|
| *detourId* | The Detour id |
| *cause* | The cause (a short, informative description) of the reason for the disruption. |
| *estimatedDelay* | The estimated delay (in minutes) |
| *affectedStops* | The affected Stops |

## 5.33.3 Member Data Documentation

### 5.33.3.1 String cause [private]

A text-based description of the cause of the Detour, intended to be displayed to customers.

If null or blank, the cause is considered Unspecified or Unknown.

### 5.33.3.2 String detourId [private]

A unique identifier for this Detour.

This is not intended to be human-readable and can be generated.

As a Detour is part of the Route Aggregate, this identifier needs only be unique within the context of a Route.

#### 5.33.3.3 int estimatedDelay [private]

The estimated time (in minutes) that each of the Stops in the.

affectedStops will be delayed.

## 5.34 Fare Class Reference

An immutable Value Object representing the cost, or 'fare,' required to ride a TransitVehicle on a particular Route.

### Public Member Functions

- Fare (BigDecimal regularFare)

  *Instantiates a new Fare with a regular fare and no discounted fair.*
- Fare (BigDecimal regularFare, BigDecimal discountedFare)

  *Instantiates a new Fare with both a regular fare and a discounted fare.*
- BigDecimal **getRegularFare** ()
- void setRegularFare (BigDecimal regularFare)
- BigDecimal **getDiscountedFare** ()
- void setDiscountedFare (BigDecimal discountedFare)

### Private Attributes

- BigDecimal regularFare

  *The normally applied fare.*
- BigDecimal discountedFare

  *A discounted fare for children, elderly, holidays, or other adjustment criteria as supplied by the TransitProvider.*

### 5.34.1 Detailed Description

An immutable Value Object representing the cost, or 'fare,' required to ride a TransitVehicle on a particular Route.

Although BusBuddy does not specify explicit rules governing the regularFare and the discountedFare, each TransitProvider must publicly state its Fare policies on a website or other public forum.

**Invariant**

if discountedFare exists, discountedFare <= regularFare

### 5.34.2 Constructor & Destructor Documentation

#### 5.34.2.1 Fare ( BigDecimal *regularFare* )

Instantiates a new Fare with a regular fare and no discounted fair.

**Precondition**

**regularFare** >= 0

**Parameters**

| | |
|---|---|
| *regularFare* | The Regular Fare |

#### 5.34.2.2 Fare ( BigDecimal *regularFare,* BigDecimal *discountedFare* )

Instantiates a new Fare with both a regular fare and a discounted fare.

**Precondition**

**regularFare** >= 0
**discountedFare** >= 0
**discountedFare** <= **discountedFare**

**Parameters**

| | |
|---|---|
| *regularFare* | The Regular Fare |
| *discountedFare* | The Discounted Fare |

### 5.34.3 Member Function Documentation

#### 5.34.3.1 void setDiscountedFare ( BigDecimal *discountedFare* )

**Precondition**

**discountedFare** >= 0 AND **discountedFare** <= regularFare if regularFare exists.

#### 5.34.3.2 void setRegularFare ( BigDecimal *regularFare* )

**Precondition**

**regularFare** >= 0

### 5.34.4 Member Data Documentation

#### 5.34.4.1 BigDecimal discountedFare [private]

A discounted fare for children, elderly, holidays, or other adjustment criteria as supplied by the TransitProvider.

This is optional. TransitProviders are not required to provide discounted Fares for any reason.

## 5.35 FavoriteTransitService Class Reference

This class is a single transit service in a user's list of favorites.

### Public Member Functions

- FavoriteTransitService (String transitServiceUrl)

  *This creates a new favorites entry.*
- String getTransitServiceUrl ()

  *This retrieves the transit service URL associated with the transit service represented by this object.*
- boolean isFavoriteTransitService ()

  *This retrieves whether this transit service itself is a favorite.*

- void setFavoriteTransitService (boolean favoriteTransitService)

  *This sets whether this transit service itself is a favorite.*
- List< String > getFavoriteRouteIds ()

  *This retrieves the ordered list of favorite routes for this transit service.*
- void setFavoriteRouteIds (List< String > favoriteRouteIds)

  *This sets the ordered list of favorite routes for this transit service.*

### Private Attributes

- final String **transitServiceUrl**
- boolean **favoriteTransitService**
- List< String > **favoriteRouteIds**

## 5.35.1 Detailed Description

This class is a single transit service in a user's list of favorites.

If just the routes are favorites, and not the service, then the favoriteTransitService boolean can be set to false. Likewise, if just the transit service is a favorite, and there are no favorite routes, the list of favorite routes can be empty.

## 5.35.2 Constructor & Destructor Documentation

### 5.35.2.1 FavoriteTransitService ( String *transitServiceUrl* )

This creates a new favorites entry.

Once created, the service URL cannot be changed.

**Parameters**

| | |
|---|---|
| *transitServiceUrl* | URL to the transit service represented by this object |

## 5.35.3 Member Function Documentation

### 5.35.3.1 List<String> getFavoriteRouteIds ( )

This retrieves the ordered list of favorite routes for this transit service.

**Returns**

ordered list of favorite route IDs

### 5.35.3.2 String getTransitServiceUrl ( )

This retrieves the transit service URL associated with the transit service represented by this object.

**Returns**

transit service URL

### 5.35.3.3 boolean isFavoriteTransitService ( )

This retrieves whether this transit service itself is a favorite.

**Returns**

true if it is, false if it is just a container object for favorite routes

#### 5.35.3.4 void setFavoriteRouteIds ( List< String > *favoriteRouteIds* )

This sets the ordered list of favorite routes for this transit service.

**Parameters**

| | |
|---|---|
| *favoriteRouteIds* | ordered list of favorite route IDs |

#### 5.35.3.5 void setFavoriteTransitService ( boolean *favoriteTransitService* )

This sets whether this transit service itself is a favorite.

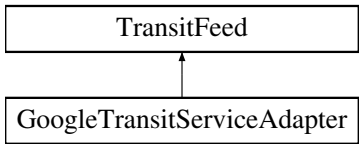
**Parameters**

| | |
|---|---|
| *favoriteTransit-Service* | true if it is, false if it is just a container object for favorite routes |

## 5.36 GoogleTransitServiceAdapter Class Reference

An Adapter Class to allow a GoogleTransitServiceAPI service to appear as a TransitService.

Inheritance diagram for GoogleTransitServiceAdapter:




### Public Member Functions

- GoogleTransitServiceAdapter (GoogleTransitServiceAPI googleTransitServiceAPI)

  *Instantiates a new GoogleTransitServiceAdapter with a GoogleTransitServiceAPI to delegate calls to.*
- Route getRoute (String routeId)

  *Gets a Route by its unique identifier.*
- Set< Route > getRoutes (Location pickup, Location dropoff, int distance)

  *Gets all available Routes that match a **pickup** or **dropoff** Location by not more than a given **distance**.*
- GoogleTransitServiceAPI **getGoogleTransitServiceAPI** ()
- void **setGoogleTransitServiceAPI** (GoogleTransitServiceAPI googleTransitServiceAPI)

### Private Attributes

- GoogleTransitServiceAPI googleTransitServiceAPI

  *The GoogleTransitServiceAPI to adapt as a TransitService.*

### 5.36.1 Detailed Description

An Adapter Class to allow a GoogleTransitServiceAPI service to appear as a TransitService.

### 5.36.2 Constructor & Destructor Documentation

#### 5.36.2.1 GoogleTransitServiceAdapter ( GoogleTransitServiceAPI *googleTransitServiceAPI* )

Instantiates a new GoogleTransitServiceAdapter with a GoogleTransitServiceAPI to delegate calls to.

**Parameters**

| | |
|---|---|
| *googleTransit-ServiceAPI* | The GoogleTransitServiceAPI to adapt. |

### 5.36.3 Member Function Documentation

#### 5.36.3.1 Route getRoute ( String *routeId* )

Gets a Route by its unique identifier.

**Precondition**

**routeId** is not null or blank.

**Postcondition**

The Route is returned if the **routeId** is found, else null.

**Parameters**

| | |
|---|---|
| *routeId* | The unique identifier of the Route |

**Returns**

The matching Route, or null if not found

Implements TransitFeed.

#### 5.36.3.2 Set<Route> getRoutes ( Location *pickup,* Location *dropoff,* int *distance* )

Gets all available Routes that match a **pickup** or **dropoff** Location by not more than a given **distance**.

**Precondition**

**pickup** is not null or blank.
**dropoff** is not null or blank.
**distance** is non-negative.

**Parameters**

| | |
|---|---|
| *pickup* | The requested pickup Location |
| *pickup* | The requested dropoff Location |
| *distance* | The distance (in miles) that each Route can deviate from the requested **pickup** or **dropoff** Location. For each Route returned, neither its start or end Location can differ from the requested **pickup** or **dropoff** Location by more than the value of the **distance** parameter. |

**Returns**

The matching Routes

Implements TransitFeed.

## 5.37 GoogleTransitServiceAPI Interface Reference

A client to Google's `Maps API`.

### 5.37.1 Detailed Description

A client to Google's `Maps API`.

Note: Actual design of the appropriate interfaces in this class are deferred to implementation teams or a further phase of design.

## 5.38 GPSLocationObject Class Reference

GPS Location is a value object used for GPS coordinates and the time of the last update.

### Public Member Functions

- double **getLatitude** ()
- void **setLatitude** (double latitude)
- double **getLongitude** ()
- void **setLongitude** (double longitude)
- Date **getLastUpdateTime** ()
- void **setLastUpdateTime** (Date lastUpdateTime)

### Private Attributes

- double latitude

  *current GPS latitude*
- double longitude

  *current GPS longitude*
- Date lastUpdateTime

  *time of last GPS update from device*

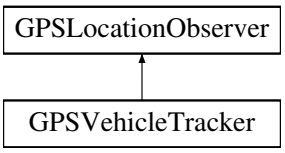
### 5.38.1 Detailed Description

GPS Location is a value object used for GPS coordinates and the time of the last update.

## 5.39 GPSLocationObserver Class Reference

Observer Pattern - Observer interface for GPS location tracking.

Inheritance diagram for GPSLocationObserver:




### Protected Member Functions

- void setGPSLocation (GPSLocationObject gpsLocation)

  *Set the current GPS location of a vehicle (state).*

### Protected Attributes

- GPSLocationTracking gpsDevice

  *Observer Pattern Subject.*
- int gpsID

  *GPS Device ID being tracked.*
- GPSLocationObject gpsLocation

  *Current GPS latitude and longitude from GPS tracker.*

### Package Functions

- abstract void gpsUpdate (int gpsID, GPSLocationObject newLocation)

  *Observer Pattern update method to update transit vehicle GPS location.*
- GPSLocationObject getGPSLocation ()

  *Return current GPS location received from a vehicle.*

## 5.39.1 Detailed Description

Observer Pattern - Observer interface for GPS location tracking.

## 5.39.2 Member Function Documentation

### 5.39.2.1 GPSLocationObject getGPSLocation ( ) [package]

Return current GPS location received from a vehicle.

This is the state of the observer pattern.

**Returns**

- Location

### 5.39.2.2 abstract void gpsUpdate ( int *gpsID,* GPSLocationObject *newLocation* ) [package],[pure virtual]

Observer Pattern update method to update transit vehicle GPS location.

**Parameters**

| | |
|---|---|
| *gpsID* | - integer device ID from the GPS unit being tracked |
| *latitude* | - double new latitude from GPS device |
| *longitude* | - double new longitude from GPS device |

Implemented in GPSVehicleTracker.

#### 5.39.2.3 void setGPSLocation ( GPSLocationObject *gpsLocation* ) [protected]

Set the current GPS location of a vehicle (state).

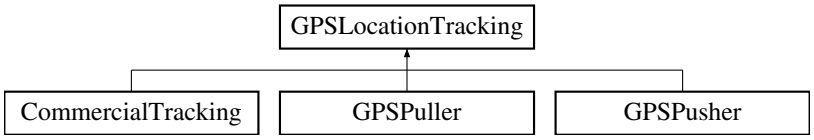
**Parameters**

| | |
|---|---|
| *gpsLocation* | - Location latest latitude and longitude of vehicle |

## 5.40 GPSLocationTracking Class Reference

GPSLocationTracking - interface Subject of the Observer Pattern Defines methods for an observer GPS Device to register and receive updates on vehicle location.

Inheritance diagram for GPSLocationTracking:




### Package Functions

- abstract void registerGPSDevice (GPSLocationObserver gpsObs)

  *registerGPSDevice - register a GPS device with the Location Tracking Service*
- abstract void unregisterGPSDevice (GPSLocationObserver gpsObs)

  *unregisterGPSDevice - remove a vehicle from list.*
- abstract void pollGPSDevice ()

  *pollGPSDevice - continuously poll registered GPS Devices for location updates*

### 5.40.1 Detailed Description

GPSLocationTracking - interface Subject of the Observer Pattern Defines methods for an observer GPS Device to register and receive updates on vehicle location.

### 5.40.2 Member Function Documentation

#### 5.40.2.1 abstract void registerGPSDevice ( GPSLocationObserver *gpsObs* ) [package], [pure virtual]

registerGPSDevice - register a GPS device with the Location Tracking Service

**Parameters**

| | |
|---|---|
| *GPSLocation-Observer* | - Vehicle location to notify when new vehicle GPS location is received |

Implemented in GPSPusher, CommercialTracking, and GPSPuller.

#### 5.40.2.2 abstract void unregisterGPSDevice ( GPSLocationObserver *gpsObs* ) [package],[pure virtual]

unregisterGPSDevice - remove a vehicle from list.

Stop updating vehicle location.

**Parameters**

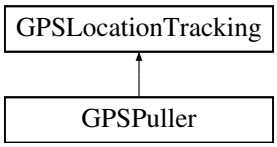
| | |
|---|---|
| *gpsObs* | GPSLocationObserver - vehicle to remove from notification list |

Implemented in GPSPusher, CommercialTracking, and GPSPuller.

## 5.41 GPSPuller Class Reference

GPS Puller is a concrete implementation of GPSLocationTracking for obtaining coordinates directly from a GPS device installed in a registered vehicle.

Inheritance diagram for GPSPuller:




### Classes

- class GPSPullerHolder

  *GPS Puller Holder is loaded on the first execution of GPSPuller.getInstance() or the first access to GPSPuller.INSTANCE, not before (lazy instantiation).*

### Package Functions

- void registerGPSDevice (GPSLocationObserver gpsObs)

  *Register a GPS Device to the list to poll for updates.*
- void unregisterGPSDevice (GPSLocationObserver gpsObs)

  *Remove a GPS device from the list currently being polled for updates.*
- void pollGPSDevice ()

  *Continuously poll the registered GPS devices for location updates.*

### Static Package Functions

- static GPSPuller **getInstance** ()

### Private Member Functions

- GPSPuller ()

  *Only need one GPS Puller Service running to track by polling all registered GPS devices.*

### Private Attributes

- ArrayList< GPSLocationObserver > gpsObserver

  *Array list of GPS devices registered for updates.*

### 5.41.1 Detailed Description

GPS Puller is a concrete implementation of GPSLocationTracking for obtaining coordinates directly from a GPS device installed in a registered vehicle.

GPS Puller is implemented as a singleton to limit the number of system resources consumed. GPS Puller uses the system infrastructure to establish a wireless network connection to the physical GPS device and retrieve update coordinates. The necessary information to contact the device is provided through the user interface when a vehicle is registered to a route.

**Postcondition**

New GPS Puller created if one did not previously exist.

### 5.41.2 Constructor & Destructor Documentation

#### 5.41.2.1 GPSPuller ( ) [private]

Only need one GPS Puller Service running to track by polling all registered GPS devices.

Constructor, creates ArrayList<GPSLocationObserver> to hold registered observers.

## 5.42 GPSPuller.GPSPullerHolder Class Reference

GPS Puller Holder is loaded on the first execution of GPSPuller.getInstance() or the first access to GPSPuller.INSTANCE, not before (lazy instantiation).

### Static Public Attributes

- static final GPSPuller **INSTANCE** = new GPSPuller()

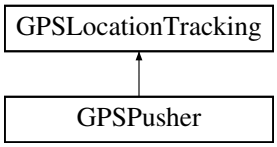
### 5.42.1 Detailed Description

GPS Puller Holder is loaded on the first execution of GPSPuller.getInstance() or the first access to GPSPuller.INSTANCE, not before (lazy instantiation).

## 5.43 GPSPusher Class Reference

Implements Subject GPSLocationTracking for retrieving GPS location updates from registered vehicles.

Inheritance diagram for GPSPusher:




### Classes

- class GPSPusherHolder

  *GPS Pusher Holder is loaded on the first execution of GPSPusher.getInstance() or the first access to GPSPusher.INSTANCE, not before (lazy instantiation).*

### Public Member Functions

- void registerGPSDevice (GPSLocationObserver gpsObs)

  *Register a GPS Device to the list to poll for updates.*
- void unregisterGPSDevice (GPSLocationObserver gpsObs)

  *Remove a GPS device from the list currently being polled for updates.*
- void pollGPSDevice ()

  *Continuously poll the registered GPS devices for location updates.*

### Static Public Member Functions

- static GPSPusher getInstance ()

  *Create a single instance of the GPS Listener for receiving GPS updates from devices that periodically push updated directly from the device.*

### Private Member Functions

- GPSPusher ()

  *Only need one GPS Pusher Service running to track by polling all registered GPS devices.*

### Private Attributes

- ArrayList< GPSLocationObserver > gpsObserver

  *Array list of GPS devices registered for updates.*

## 5.43.1 Detailed Description

Implements Subject GPSLocationTracking for retrieving GPS location updates from registered vehicles.

GPSPusher uses system infrastructure resources to set up a network listener to receive updates directly from the GPS device. GPS Pusher is implemented as a singleton to limit the number of system resources consumed. GPS Pusher receives the necessary configuration information (e.g. port) from the user interface when the GPS device is registered.

**Postcondition**

New GPS Listener created if none existed previously.

## 5.43.2 Constructor & Destructor Documentation

### 5.43.2.1 GPSPusher ( ) [private]

Only need one GPS Pusher Service running to track by polling all registered GPS devices.

Constructor, creates ArrayList<GPSLocationObserver> to hold registered observers. < List of GPS devices currently registered and waiting for updates

## 5.43.3 Member Function Documentation

### 5.43.3.1 static GPSPusher getInstance ( ) [static]

Create a single instance of the GPS Listener for receiving GPS updates from devices that periodically push updated directly from the device.

**Returns**

GPSPusher reference to the listener for incoming GPS updates from registered devices.

## 5.44 GPSPusher.GPSPusherHolder Class Reference

GPS Pusher Holder is loaded on the first execution of GPSPusher.getInstance() or the first access to GPSPusher.-INSTANCE, not before (lazy instantiation).

### Static Public Attributes

- static final GPSPusher **INSTANCE** = new GPSPusher()

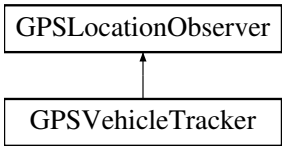
### 5.44.1 Detailed Description

GPS Pusher Holder is loaded on the first execution of GPSPusher.getInstance() or the first access to GPSPusher.-INSTANCE, not before (lazy instantiation).

## 5.45 GPSVehicleTracker Class Reference

Implementation of the Observer, update transit vehicle GPS location and time GPS Vehicle Tracker gets the state as new GPS coordinates and time from GPS Location Tracking and updates the transit vehicle.

Inheritance diagram for GPSVehicleTracker:




### Public Member Functions

- GPSVehicleTracker (GPSLocationTracking gpsDevice)

  *Register the Transit Vehicle GPS device with GPS location tracking.*
- void gpsUpdate (int gpsID, GPSLocationObject newLocation)

  *Notify method to get the new GPS coordinates from GPS location tracking.*

### Additional Inherited Members

### 5.45.1 Detailed Description

Implementation of the Observer, update transit vehicle GPS location and time GPS Vehicle Tracker gets the state as new GPS coordinates and time from GPS Location Tracking and updates the transit vehicle.

**Postcondition**

Transit Vehicle GPS location updated.

### 5.45.2 Constructor & Destructor Documentation

#### 5.45.2.1 GPSVehicleTracker ( GPSLocationTracking *gpsDevice* )

Register the Transit Vehicle GPS device with GPS location tracking.

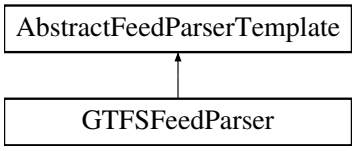
**Parameters**

| | |
|---|---|
| *gpsDevice* | - GPSLocationTracking Subject being observed |

## 5.46 GTFSFeedParser Class Reference

A AbstractFeedParserTemplate implementation designed to parse GTFS format ZIP files into Routes.

Inheritance diagram for GTFSFeedParser:




### Protected Member Functions

- Set< Route > parseFeed (InputStream feed)

  *Parse the GTFS format ZIP files into Routes.*

### Additional Inherited Members

### 5.46.1 Detailed Description

A AbstractFeedParserTemplate implementation designed to parse GTFS format ZIP files into Routes.

### 5.46.2 Member Function Documentation

#### 5.46.2.1 Set<Route> parseFeed ( InputStream *feed* ) [protected], [virtual]

Parse the GTFS format ZIP files into Routes.

**See Also**

AbstractFeedParserTemplate::parseFeed

Implements AbstractFeedParserTemplate.

## 5.47 HashUtility Class Reference

This is a utility class to handle secure hashes.

### Static Public Member Functions

- static String hash (String input)

  *This is a method that will take an input string, securely hash it, and return the hashed String using the SHA-512 algorithm.*

### 5.47.1 Detailed Description

This is a utility class to handle secure hashes.

### 5.47.2 Member Function Documentation

#### 5.47.2.1 static String hash ( String *input* ) [static]

This is a method that will take an input string, securely hash it, and return the hashed String using the SHA-512 algorithm.

**Parameters**

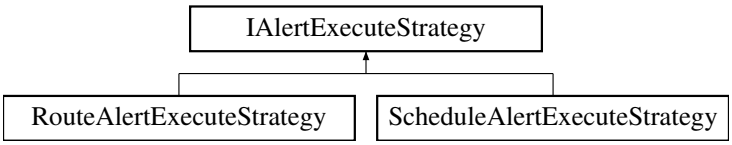
| | |
|---|---|
| *input* | |

**Returns**

## 5.48 IAlertExecuteStrategy Interface Reference

An interface for executing different type of alert based on alert type.

Inheritance diagram for IAlertExecuteStrategy:




### Public Member Functions

- boolean execute (Alert alertModel)

  *A method that executes alert passed in based on the type of alert.*

### 5.48.1 Detailed Description

An interface for executing different type of alert based on alert type.

Each implementation of this interface will apply their own execute method that can send alert based on user, route etc.

### 5.48.2 Member Function Documentation

#### 5.48.2.1 boolean execute ( Alert *alertModel* )

A method that executes alert passed in based on the type of alert.

**Parameters**

| | |
|---|---|
| *alertModel* | An Alert model fetched from database. |

**Returns**

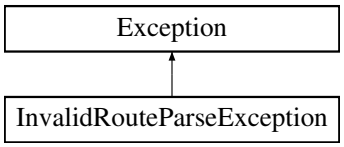
A boolean indicating if overall process complete successfully.

Implemented in RouteAlertExecuteStrategy, and ScheduleAlertExecuteStrategy.

## 5.49 InvalidRouteParseException Class Reference

An InvalidRouteParseException indicates an invalid batch of parsed Routes has been been detected.

Inheritance diagram for InvalidRouteParseException:




### Public Member Functions

- Set< Route > **getRouteBatch** ()
- void **setRouteBatch** (Set< Route > routeBatch)

### Protected Member Functions

- InvalidRouteParseException (Set< Route > routeBatch)

  *Instantiates a new invalid route parse exception.*

### Private Attributes

- Set< Route > routeBatch

  *The failed Route batch.*

### Static Private Attributes

- static final long serialVersionUID = -4399874766965916500L

  *The Constant serialVersionUID.*

### 5.49.1 Detailed Description

An InvalidRouteParseException indicates an invalid batch of parsed Routes has been been detected.

Note that one ore more of the referenced Routes are invalid, but not necessarily all of them are invalid.

### 5.49.2 Constructor & Destructor Documentation

#### 5.49.2.1 InvalidRouteParseException ( Set< Route > *routeBatch* ) [protected]

Instantiates a new invalid route parse exception.

**Parameters**

| | |
|---|---|
| *routeBatch* | the route batch |

### 5.49.3 Member Data Documentation

#### 5.49.3.1 Set<Route> routeBatch [private]

The failed Route batch.

Handlers of this Exception may choose to re-validate, fix, and/or retry the operation with an adjusted batch.

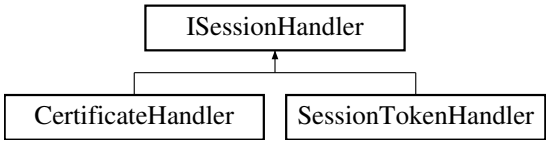
#### 5.49.3.2 final long serialVersionUID = -4399874766965916500L [static], [private]

The Constant serialVersionUID.

## 5.50 ISessionHandler Interface Reference

An interface to verify the validity of encrypted token being passed.

Inheritance diagram for ISessionHandler:




### Public Member Functions

- boolean verifySessionToken (String encryptedToken) throws BusBuddyForbiddenException

  *This method takes in an encrypted token in a form of string and then validates for its authenticity.*

### 5.50.1 Detailed Description

An interface to verify the validity of encrypted token being passed.

The implementation of this interface must decrpyt the session token or certificate and then verify it.

### 5.50.2 Member Function Documentation

#### 5.50.2.1 boolean verifySessionToken ( String *encryptedToken* ) throws BusBuddyForbiddenException

This method takes in an encrypted token in a form of string and then validates for its authenticity.

**Parameters**

| | |
|---|---|
| *encryptedToken* | • An encrypted string. |

**Returns**

A boolean indicating whether the token was valid or not.

**Exceptions**

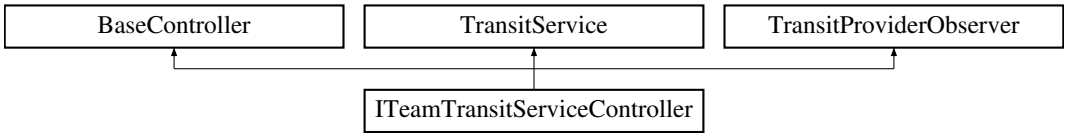
| | |
|---|---|
| *BusBuddyForbidden-Exception* | • Throws an exception when the token is not valid. |

Implemented in SessionTokenHandler, and CertificateHandler.

## 5.51 ITeamTransitServiceController Class Reference

The iTeam implementation of the TransitService that exposes Transit data via a REST Service.

Inheritance diagram for ITeamTransitServiceController:




### Public Member Functions

- void handleRouteDisruptionEvent (RouteDisruptionEvent routeDisruptionEvent)

  *After a RouteDisruptionEvent is received, this class will perform the following:*
- Route getRoute (String routeId)

  *Gets a Route by its unique identifier.*
- Set< Route > getRoutes (Location pickup, Location dropoff, int distance)

  *Gets all available Routes that match a **pickup** or **dropoff** Location by not more than a given **distance**.*
- TransitInfo getTransitInfo ()

  *Gets metadata about the Transit Authority providing the information retrieved from this service.*
- URL getServiceURL ()

  *The URL that uniquely identifies this TransitService.*
- TransitFeed **getTransitFeed** ()
- void **setTransitFeed** (TransitFeed transitFeed)
- AlertRequestController **getAlertRequestController** ()
- void **setAlertRequestController** (AlertRequestController alertRequestController)

### Private Attributes

- TransitFeed transitFeed

  *The TransitFeed used to provide data to this TransitService implementation.*
- AlertRequestController alertRequestController

  *The inter-module dependency to the Alert Module.*

### 5.51.1 Detailed Description

The iTeam implementation of the TransitService that exposes Transit data via a REST Service.

Note that this is intended to receive and respond to HTTP requests via JSON. The actual Object -> JSON transformation, as well as the specification of HTTP REST URLs is left to the development phase and is not specified here.

Errors should be communicated via standard HTTP error codes. The BaseController will be responsible for mapping Exception types to HTTP error codes, so actual error codes are not specified explicitly here.

### 5.51.2 Member Function Documentation

#### 5.51.2.1 Route getRoute ( String *routeId* )

Gets a Route by its unique identifier.

**Parameters**

| | |
|---|---|
| *routeId* | The unique identifier of the Route |

**Returns**

The matching Route

**Precondition**

**routeId** is not null or blank.

**Postcondition**

The Route is returned if the **routeId** is found, else null.

**Exceptions**

| | |
|---|---|
| *BusBuddyNotFound-Exception* | Thrown if the Route with the given **routeId** is not found. |

Implements TransitService.

#### 5.51.2.2 Set<Route> getRoutes ( Location *pickup,* Location *dropoff,* int *distance* )

Gets all available Routes that match a **pickup** or **dropoff** Location by not more than a given **distance**.

**Parameters**

| | |
|---|---|
| *pickup* | The requested dropoff Location |
| *dropoff* | the dropoff |
| *distance* | The distance (in miles) that each Route can deviate from the requested **pickup** or **dropoff** |

**Returns**

The matching Routes

**Precondition**

**pickup** is not null or blank.
**dropoff** is not null or blank.
**distance** is non-negative. Location. For each Route returned, neither its start or end Location can differ from the requested **pickup** or **dropoff** Location by more than the value of the **distance** parameter.

Implements TransitService.

#### 5.51.2.3 URL getServiceURL ( )

The URL that uniquely identifies this TransitService.

In a REST environment, this might be the root of the REST API path.

**Returns**

The URL of this service

Implements TransitService.

#### 5.51.2.4 TransitInfo getTransitInfo ( )

Gets metadata about the Transit Authority providing the information retrieved from this service.

**Returns**

The TransitInfo of the Transit Authority of this service.

Implements TransitService.

#### 5.51.2.5 void handleRouteDisruptionEvent ( RouteDisruptionEvent *routeDisruptionEvent* )

After a RouteDisruptionEvent is received, this class will perform the following:

- Internally register the RouteDisruptionEvent so that any subsequent requests for affected Routes will include appropriate Detour information.
- Notify the Alert module via the AlertRequestController of the disruption with the updated Route. This updated Route should include all necessary Detour information.

Implements TransitProviderObserver.

### 5.51.3 Member Data Documentation

#### 5.51.3.1 AlertRequestController alertRequestController [private]

The inter-module dependency to the Alert Module.

The AlertRequestController accepts requests from this class to inform the Alert Module of a RouteDisruptionAlert.

NOTE: This represents a conceptual dependency to the AlertRequestController. During implementation phase, actual communication with the AlertRequestController will happen via some client object or service. Implementation of the actual client and its link to the AlertRequestController (to include REST URLS and JSON structure) will be left to the next phase, or as design details for the development team.

#### 5.51.3.2 TransitFeed transitFeed [private]

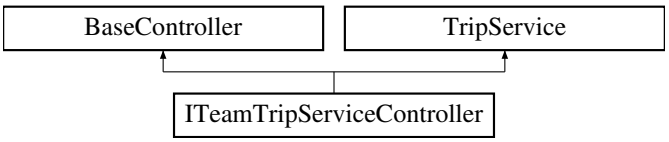
The TransitFeed used to provide data to this TransitService implementation.

Note that this TransitFeed implementation may be aggregate of many TransitFeeds, an Adapter to another API, or other implementation.

## 5.52 ITeamTripServiceController Class Reference

An iTeam implementation of the TripService that exposes Trip data via a REST Service.

Inheritance diagram for ITeamTripServiceController:




### Public Member Functions

- Trip calculateTrip (Location start, Location end)

  *Calculate an optimal Trip given a **start** Location and an **end** Location.*
- TransitService **getTransitService** ()
- void **setTransitService** (TransitService transitService)

### Private Attributes

- TransitService transitService

  *The TransitService used to provide the Route data used in the Trip calculations.*

### 5.52.1 Detailed Description

An iTeam implementation of the TripService that exposes Trip data via a REST Service.

Note: The actual Trip calculation algorithm is not specified here and is beyond the scope of this project.

Errors should be communicated via standard HTTP error codes. The BaseController will be responsible for mapping Exception types to HTTP error codes, so actual error codes are not specified explicitly here.

### 5.52.2 Member Function Documentation

#### 5.52.2.1 Trip calculateTrip ( Location *start*, Location *end* )

Calculate an optimal Trip given a **start** Location and an **end** Location.

**Parameters**

| | |
|---|---|
| *start* | The requested start Location of the Trip. |
| *end* | The requested end Location of the Trip. |

**Returns**

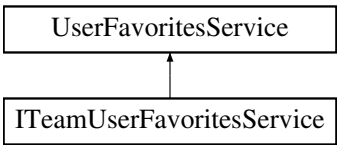
The calculated Trip

Implements TripService.

## 5.53 ITeamUserFavoritesService Class Reference

This is the iTeam's implementation of UserFavoritesService.

Inheritance diagram for ITeamUserFavoritesService:




### Public Member Functions

- UserFavoritesList readFavorites (String sessionToken) throws BusBuddyException
- void saveFavorites (String sessionToken, UserFavoritesList favorites) throws BusBuddyException

### Protected Attributes

- ITeamUserLoginService **userLoginService**
- UserFavoritesRepository **userFavoritesRepository**

### 5.53.1 Detailed Description

This is the iTeam's implementation of UserFavoritesService.

### 5.53.2 Member Function Documentation

#### 5.53.2.1 UserFavoritesList readFavorites ( String *sessionToken* ) throws BusBuddyException

**See Also**

UserFavoritesService.readFavorites

Implements UserFavoritesService.

#### 5.53.2.2 void saveFavorites ( String *sessionToken,* UserFavoritesList *favorites* ) throws BusBuddyException

**See Also**

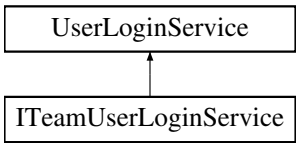
UserFavoritesService.saveFavorites

Implements UserFavoritesService.

## 5.54 ITeamUserLoginService Class Reference

This is the iTeam's implementation of UserLoginService.

Inheritance diagram for ITeamUserLoginService:




### Public Member Functions

- String login (String username, String password) throws BusBuddyException
- void logout (String sessionToken) throws BusBuddyException
- User getUser (String sessionToken) throws BusBuddyException
- String createAlertSession (String sessionToken) throws BusBuddyException
- void sendUsername (String email) throws BusBuddyException
- void sendUsername (short countryCode, String mobile) throws BusBuddyException
- void resetPassword (String username, String email) throws BusBuddyException
- void resetPassword (String username, short countryCode, String mobile) throws BusBuddyException

### Protected Member Functions

- boolean checkPermissions (String sessionToken, User user) throws BusBuddyException

  *This method checks to see if the currently logged in user has permissions to modify another given user object.*

### Protected Attributes

- SessionRepository **sessionRepository**
- UserRepository **userRepository**

### 5.54.1 Detailed Description

This is the iTeam's implementation of UserLoginService.

### 5.54.2 Member Function Documentation

#### 5.54.2.1 boolean checkPermissions ( String *sessionToken,* User *user* ) throws BusBuddyException [protected]

This method checks to see if the currently logged in user has permissions to modify another given user object.

A session can modify a user if it is the currently logged in user that is being modified, or if the currently logged in user is a system administrator.

**Precondition**

The session token must be linked to an active and valid session, which must be linked to an active account.

**Postcondition**

The expiration time will be advanced based on this activity against the session.

**Parameters**

| | |
|---|---|
| *sessionToken* | The session token identifying the session of the currently logged in user. |
| *user* | The user we are checking to see if the session has permission to modify. |

**Returns**

true if the currently signed in user has permission to modify the user specified in the user parameter, false otherwise

**Exceptions**

| | |
|---|---|
| *BusBuddyBadRequestException* | This exception is thrown if the session token is blank. |
| *BusBuddyForbiddenException* | This exception is thrown if the session token is invalid, linked to an expired session, or the user does not have permission to be signed in. |
| *BusBuddyInternalException* | This exception is thrown if an internal error prevents processing of the request. |

#### 5.54.2.2 String createAlertSession ( String *sessionToken* ) throws BusBuddyException

**See Also**

UserLoginService.createAlertSession

Implements UserLoginService.

#### 5.54.2.3 User getUser ( String *sessionToken* ) throws BusBuddyException

**See Also**

UserLoginService.getUser

Implements UserLoginService.

#### 5.54.2.4 String login ( String *username,* String *password* ) throws BusBuddyException

**See Also**

UserLoginService.login

Implements UserLoginService.

#### 5.54.2.5 void logout ( String *sessionToken* ) throws BusBuddyException

**See Also**

UserLoginService.logout

Implements UserLoginService.

#### 5.54.2.6 void resetPassword ( String *username,* String *email* ) throws BusBuddyException

**See Also**

UserLoginService.resetPassword(String, String)

Implements UserLoginService.

#### 5.54.2.7 void resetPassword ( String *username,* short *countryCode,* String *mobile* ) throws BusBuddyException

**See Also**

UserLoginService.resetPassword(String, short, String)

Implements UserLoginService.

#### 5.54.2.8 void sendUsername ( String *email* ) throws BusBuddyException

**See Also**

UserLoginService.sendUsername(String)

Implements UserLoginService.

#### 5.54.2.9 void sendUsername ( short *countryCode,* String *mobile* ) throws BusBuddyException

**See Also**

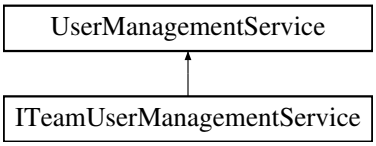
UserLoginService.sendUsername(short, String)

Implements UserLoginService.

## 5.55 ITeamUserManagementService Class Reference

This is the iTeam's implementation of UserManagementService.

Inheritance diagram for ITeamUserManagementService:




### Public Member Functions

- User createUser (User userToCreate, String password) throws BusBuddyException
- User findUserByUsername (String sessionToken, String username) throws BusBuddyException
- User findUserByEmail (String sessionToken, String email) throws BusBuddyException
- User findUserByMobile (String sessionToken, short countryCode, String mobile) throws BusBuddyException
- void updateUser (String sessionToken, User newUserData, String password) throws BusBuddyException
- void deleteUser (String sessionToken, User userToDelete) throws BusBuddyException

### Protected Attributes

- ITeamUserLoginService **userLoginService**
- UserRepository **userRepository**

### 5.55.1 Detailed Description

This is the iTeam's implementation of UserManagementService.

### 5.55.2 Member Function Documentation

#### 5.55.2.1 User createUser ( User *userToCreate,* String *password* ) throws BusBuddyException

**See Also**

UserManagementService.createUser

Implements UserManagementService.

#### 5.55.2.2 void deleteUser ( String *sessionToken,* User *userToDelete* ) throws BusBuddyException

**See Also**

UserManagementService.delete

Implements UserManagementService.

#### 5.55.2.3 User findUserByEmail ( String *sessionToken,* String *email* ) throws BusBuddyException

**See Also**

UserManagementService.findUserByEmail

Implements UserManagementService.

#### 5.55.2.4 User findUserByMobile ( String *sessionToken,* short *countryCode,* String *mobile* ) throws BusBuddyException

**See Also**

UserManagementService.findUserByMobile

Implements UserManagementService.

#### 5.55.2.5 User findUserByUsername ( String *sessionToken,* String *username* ) throws BusBuddyException

**See Also**

UserManagementService.findUserByUsername

Implements UserManagementService.

#### 5.55.2.6 void updateUser ( String *sessionToken,* User *newUserData,* String *password* ) throws BusBuddyException

**See Also**

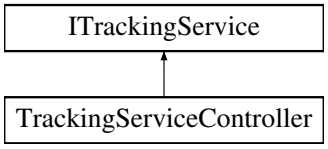
UserManagementService.updateUser

Implements UserManagementService.

## 5.56 ITrackingService Interface Reference

Interface for the Tracking Service Controller.

Inheritance diagram for ITrackingService:




## Public Member Functions

- void registerVehicleOnRoute (URL url, int gpsDeviceID)

  *Interface for registering vehicles on a route from the user interface.*
- void unregisterVehicleFromRoute (String url, int gpsDeviceID)

  *Interface for removing a registered vehicle from a route when the vehicle goes out of service.*
- void addUserTrackingAlert (UserTrackingAlertObject utao)

  *Add a user tracking alert from the user interface.*
- void startTrackingController ()

  *Logic necessary when the tracking controller is cold started.*
- GPSLocationObject getTransitVehicleLocation (int gpsDeviceID)

  *Allows users and modules outside of the tracking package to query for the current location of a registered vehicle.*

### 5.56.1 Detailed Description

Interface for the Tracking Service Controller.

The tracking service purpose is to allow vehicles to register on routes and to establish a connection with the GPS device installed in the vehicle to provide regular vehicle location updates. The tracking service uses the current GPS coordinates to determine when to send alerts to registered users.

### 5.56.2 Member Function Documentation

#### 5.56.2.1 void addUserTrackingAlert ( UserTrackingAlertObject *utao* )

Add a user tracking alert from the user interface.

**Parameters**

| | |
|---|---|
| *utao* | - UserTrackingAlertObject information from user interface necessary to create alert. |

Implemented in TrackingServiceController.

#### 5.56.2.2 GPSLocationObject getTransitVehicleLocation ( int *gpsDeviceID* )

Allows users and modules outside of the tracking package to query for the current location of a registered vehicle.

**Parameters**

| | |
|---|---|
| *gpsDeviceID* | - unique hardware GPS device |

**Returns**

- Location current latitude and longitude of vehicle

Implemented in TrackingServiceController.

#### 5.56.2.3 void registerVehicleOnRoute ( URL *url*, int *gpsDeviceID* )

Interface for registering vehicles on a route from the user interface.

**Parameters**

| | |
|---|---|
| *url* | - Transit company URL |
| *gpsDeviceID* | - unique hardware GPS device ID |

Implemented in TrackingServiceController.

#### 5.56.2.4 void startTrackingController ( )

Logic necessary when the tracking controller is cold started.

Retrieves the saved user alerts from the IAlertService

Implemented in TrackingServiceController.

#### 5.56.2.5 void unregisterVehicleFromRoute ( String *url*, int *gpsDeviceID* )

Interface for removing a registered vehicle from a route when the vehicle goes out of service.

**Parameters**

| | |
|---|---|
| *url* | - URL uniquely identifying a transit company. |
| *gpsDeviceID* | - unique hardware GPS id being unregistered on user interface |

Implemented in TrackingServiceController.

## 5.57 Location Class Reference

An immutable Value Object representing a physical point on the geographic coordinate system.

### Public Member Functions

- Location (double latitude, double longitude)

  *Instantiates a new immutable Location with the given latitude and longitude.*
- double **getLatitude** ()
- double **getLongitude** ()

### Private Attributes

- double latitude

  *The latitude of the geographic point.*
- double longitude

  *The longitude of the geographic point.*

### 5.57.1 Detailed Description

An immutable Value Object representing a physical point on the geographic coordinate system.

### 5.57.2 Constructor & Destructor Documentation

#### 5.57.2.1 Location ( double *latitude,* double *longitude* )

Instantiates a new immutable Location with the given latitude and longitude.

**Parameters**

| | |
|---|---|
| *latitude* | The point latitude |
| *longitude* | The point longitude |

## 5.58 MessageDeliveryUtility Class Reference

This is a utility class to handle message delivery, such as through email or SMS.

### Static Public Member Functions

- static void sendEmail (String to, String from, String subject, String htmlBody) throws BusBuddyInternalException

  *This method sends an HTML e-mail.*
- static void sendSms (short countryCode, String mobileNumber, String message)

  *This method sends an SMS text message.*

### 5.58.1 Detailed Description

This is a utility class to handle message delivery, such as through email or SMS.

### 5.58.2 Member Function Documentation

#### 5.58.2.1 static void sendEmail ( String *to,* String *from,* String *subject,* String *htmlBody* ) throws BusBuddyInternalException [static]

This method sends an HTML e-mail.

**Parameters**

| | |
|---|---|
| *to* | recipient address |
| *from* | sender address |
| *subject* | subject line |
| *htmlBody* | HTML body of the message |

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown if there is an error sending the e-mail. |

#### 5.58.2.2 static void sendSms ( short *countryCode,* String *mobileNumber,* String *message* ) [static]

This method sends an SMS text message.

**Precondition**

The mobile number must be a String consisting entirely of digits.

**Parameters**

| | |
|---|---|
| *countryCode* | country code for the recipient |
| *mobileNumber* | mobile number to send to |
| *message* | body of the message to send |

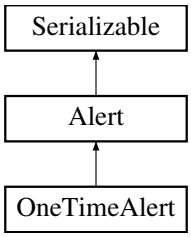
**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown if there is an error sending the message. |

## 5.59 OneTimeAlert Class Reference

This is a model of alert that is to be run one time only.

Inheritance diagram for OneTimeAlert:




### Public Member Functions

- Date **getDateExecuted** ()
- void **setDateExecuted** (Date dateExecuted)

### Private Attributes

- Date dateExecuted

  *Date when it was executed.*

### Static Private Attributes

- static final long **serialVersionUID** = 8851691556082123516L

### 5.59.1 Detailed Description

This is a model of alert that is to be run one time only.

This can be configure by User (e.g., catch bus to Boston at 9am on MM/DD/YYYY) or by any other module (e.g., Route to MSP downtown on MM/DD/YYYY is going to be rerouted). This class extends the Alert.

### 5.59.2 Member Data Documentation

#### 5.59.2.1 Date dateExecuted [private]

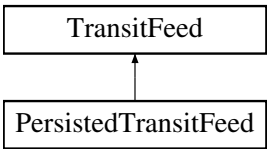
Date when it was executed.

If absent, then the alert hasn't been executed.

## 5.60 PersistedTransitFeed Class Reference

An implementation of the TransitFeed interface that communicates with a RouteRepository to retrieve its data.

Inheritance diagram for PersistedTransitFeed:




### Public Member Functions

- Route getRoute (String routeId)

  *Gets a Route by its unique identifier.*
- Set< Route > getRoutes (Location pickup, Location dropoff, int distance)

  *Gets all available Routes that match a **pickup** or **dropoff** Location by not more than a given **distance**.*
- RouteRepository **getRouteRepository** ()
- void **setRouteRepository** (RouteRepository routeRepository)

### Private Attributes

- RouteRepository routeRepository

  *The RouteRepository responsible for providing data.*

### 5.60.1 Detailed Description

An implementation of the TransitFeed interface that communicates with a RouteRepository to retrieve its data.

This implementation is appropriate when a retrieving data from a TransitProvider that does not already supply an external API that can be used at runtime. If the data needs to be parsed and imported into a RouteRepository, this implementation will expose that persisted data as a TransitFeed.

### 5.60.2 Member Function Documentation

#### 5.60.2.1 Route getRoute ( String *routeId* )

Gets a Route by its unique identifier.

**Precondition**

**routeId** is not null or blank.

**Postcondition**

The Route is returned if the **routeId** is found, else null.

**Parameters**

| | |
|---|---|
| *routeId* | The unique identifier of the Route |

**Returns**

The matching Route, or null if not found

Implements TransitFeed.

#### 5.60.2.2 Set<Route> getRoutes ( Location *pickup*, Location *dropoff*, int *distance* )

Gets all available Routes that match a **pickup** or **dropoff** Location by not more than a given **distance**.

**Precondition**

**pickup** is not null or blank.
**dropoff** is not null or blank.
**distance** is non-negative.

**Parameters**

| | |
|---|---|
| *pickup* | The requested pickup Location |
| *pickup* | The requested dropoff Location |
| *distance* | The distance (in miles) that each Route can deviate from the requested **pickup** or **dropoff** Location. For each Route returned, neither its start or end Location can differ from the requested **pickup** or **dropoff** Location by more than the value of the **distance** parameter. |

**Returns**

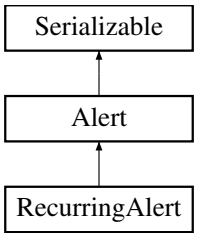
The matching Routes

Implements TransitFeed.

## 5.61 RecurringAlert Class Reference

This is a model of alert that is to be run multiple times.

Inheritance diagram for RecurringAlert:




### Public Member Functions

- Date **getSuspendDateTime** ()
- void **setSuspendDateTime** (Date suspendDateTime)
- Date **getResumeDateTime** ()
- void **setResumeDateTime** (Date resumeDateTime)
- Date **getLastSuccessfullyRanOnDateTime** ()
- void **setLastSuccessfullyRanOnDateTime** (Date lastSuccessfullyRanOnDateTime)
- List< RecurringData > **getRecurringData** ()
- void **setRecurringData** (List< RecurringData > recurringData)

- int **getRepeatEvery** ()
- void **setRepeatEvery** (int repeatEvery)
- AlertRecurringType **getAlertRecurringType** ()
- void **setAlertRecurringType** (AlertRecurringType alertRecurringType)

### Private Attributes

- Date suspendDateTime

  *DateTime when the alert is to be suspended temporarily.*
- Date resumeDateTime

  *DateTime when the alert is to be resumed.*
- Date lastSuccessfullyRanOnDateTime

  *DateTime of last successful run.*
- List< RecurringData > recurringData

  *List of RecurringData that holds the information about when the alert should actually run.*
- int repeatEvery

  *Parameter to signify the skip count.*
- AlertRecurringType alertRecurringType

  *Type of recurring alert.*

### Static Private Attributes

- static final long **serialVersionUID** = -475174398668611743L

### 5.61.1 Detailed Description

This is a model of alert that is to be run multiple times.

Depending on User or other modules, the alert will run yearly, monthly, daily in specified hour and minute.

### 5.61.2 Member Data Documentation

#### 5.61.2.1 AlertRecurringType alertRecurringType [private]

Type of recurring alert.

Value is as defined in AlertRecurringType

#### 5.61.2.2 int repeatEvery [private]

Parameter to signify the skip count.

Valid value is >0 If alert is to occur every Monday and the repeatEvery is set to 2, then it will repeat once every 2 week.

## 5.62 RecurringData Class Reference

This model holds the information about the date and time the alert needs to run.

### Public Member Functions

- int **getDayOfYear** ()
- void **setDayOfYear** (int dayOfYear)
- int **getDayOfMonth** ()
- void **setDayOfMonth** (int dayOfMonth)
- int **getDayOfWeek** ()
- void **setDayOfWeek** (int dayOfWeek)
- int **getStartHour** ()
- void **setStartHour** (int startHour)
- int **getStartMinute** ()
- void **setStartMinute** (int startMinute)

### Private Attributes

- int dayOfYear

  *Day of year that the alert should run.*
- int dayOfMonth

  *Day of month that the alert should run.*
- int dayOfWeek

  *Day of week that the alert should run.*
- int startHour

  *The exact hour when the alert should run.*
- int startMinute

  *The exact minute when the alert should run.*

## 5.62.1 Detailed Description

This model holds the information about the date and time the alert needs to run.

## 5.62.2 Member Data Documentation

### 5.62.2.1 int dayOfMonth [private]

Day of month that the alert should run.

Valid value is from 1-28.

### 5.62.2.2 int dayOfWeek [private]

Day of week that the alert should run.

e.g., 1 = Sunday and 7 = Saturday.

### 5.62.2.3 int dayOfYear [private]

Day of year that the alert should run.

Valid value = 1-365

#### 5.62.2.4 int startHour [private]

The exact hour when the alert should run.

Valid value is from 0 - 23

#### 5.62.2.5 int startMinute [private]

The exact minute when the alert should run.

Valid value is from 0-59.

## 5.63 Route Class Reference

A Route is a TransitVehicle path of travel, or a "Line," as referred to by a TransitProvider.

### Public Member Functions

- List< Stop > **getStops** ()
- void setStops (List< Stop > stops)

  *Sets the Stops of this Route.*
- String **getRouteId** ()
- void **setRouteId** (String routeId)
- String **getRouteName** ()
- void **setRouteName** (String routeName)
- Set< Detour > **getDetours** ()
- void **setDetours** (Set< Detour > detours)

### Protected Member Functions

- Route (String routeId, String routeName, List< Stop > stops)

  *Instantiates a new Route with all required fields.*

### Private Attributes

- String routeId

  *A unique identifier for this Route.*
- String routeName

  *Text to display in maps and other literature to denote this Route.*
- List< Stop > stops

  *And ordered list of Stops to be visited in this Route.*
- Set< Detour > detours

  *A set of Detours, or disruptions in Route availability and/or Stop schedule.*

### 5.63.1 Detailed Description

A Route is a TransitVehicle path of travel, or a "Line," as referred to by a TransitProvider.

A Route can be thought of as an ordered list of Stops.

Note that Routes may add/remove stops, change Stop times, or be disrupted by Detours, while still remaining the same Route.

Route is also the AggregateRoot of the RouteAggregate, which implies that the routeId parameter is globally unique.

**Invariant**

stops.size() >= 2

### 5.63.2 Constructor & Destructor Documentation

#### 5.63.2.1 Route ( String *routeId,* String *routeName,* List< Stop > *stops* ) [protected]

Instantiates a new Route with all required fields.

**Precondition**

**stops**.size() >= 2

**Parameters**

| | |
|---|---|
| *routeId* | The Route id |
| *routeName* | The Route name |
| *stops* | The Stops in this Route. |

### 5.63.3 Member Function Documentation

#### 5.63.3.1 void setStops ( List< Stop > *stops* )

Sets the Stops of this Route.

Note that there must be 2 or more Stops in a Route.

**Precondition**

**stops**.size() >= 2

### 5.63.4 Member Data Documentation

#### 5.63.4.1 Set<Detour> detours [private]

A set of Detours, or disruptions in Route availability and/or Stop schedule.

These Detours represent disruptions that are current at the time of retrieval of this Route.

#### 5.63.4.2 String routeName [private]

Text to display in maps and other literature to denote this Route.

Uniqueness is not enforced, but this name should provide enough context to allow users to distinguish this Route.

#### 5.63.4.3 List<Stop> stops [private]

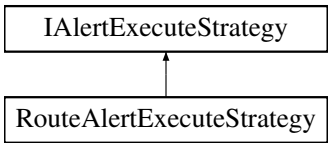
And ordered list of Stops to be visited in this Route.

Stops must be visited in order unless there is a disruption in service, in which case clients can expect a RouteDistruptionAlert and/or an Detour.

## 5.64 RouteAlertExecuteStrategy Class Reference

A concrete implementation of IAlertExecuteStrategy that handles executing alert related to route changes.

Inheritance diagram for RouteAlertExecuteStrategy:




## Public Member Functions

- boolean execute (Alert alert)

## Package Attributes

- AlertRepository alertRepository

  *An instance of AlertRepository that is used to fetch alerts that are effected by particular route.*
- AlertUserClient userClient

  *A spring autowired instance of AlertUserClient that can call the User module to get user information.*

### 5.64.1 Detailed Description

A concrete implementation of IAlertExecuteStrategy that handles executing alert related to route changes.

### 5.64.2 Member Function Documentation

#### 5.64.2.1 boolean execute ( Alert *alert* )

**See Also**

IAlertExecuteStrategy::execute(List) Takes in a list of alerts usually provided from Transit or Tracking module where route disruption information is stored. This method finds all the alerts (that users have created) that are associated with a particular route. Hence, each route disruption alert can execute multiple other alerts.

Find other alert that is associated with routeId in this alert.

call user module for each and every of these alerts and push the notification.

if success

Implements IAlertExecuteStrategy.

### 5.64.3 Member Data Documentation

#### 5.64.3.1 AlertRepository alertRepository [package]

An instance of AlertRepository that is used to fetch alerts that are effected by particular route.

This is autowired via Spring Framework.

#### 5.64.3.2 AlertUserClient userClient [package]

A spring autowired instance of AlertUserClient that can call the User module to get user information.

This is autowired via Spring Framework.

## 5.65 RouteDisruptionAlert Class Reference

An Alert indicating a disruption of normal Route availability or scheduling.

### Public Member Functions

- URL **getTransitServiceUrl** ()
- void **setTransitServiceUrl** (URL transitServiceUrl)
- String **getRouteId** ()
- void **setRouteId** (String routeId)

### Private Attributes

- URL transitServiceUrl

  *The URL callback of the originating TransitService.*
- String routeId

  *The unique identifier of the affected Route.*

### 5.65.1 Detailed Description

An Alert indicating a disruption of normal Route availability or scheduling.

Clients interested in more specific information about the disruption, including cause and affected Stops, should use the getTransitServiceUrl() method to establish a link to the appropriate TransitService, and then obtain the affected Route using the routeId from the getRouteId() method.

Once retrieved, current Detour information can be accessed via the Route getDetours() method on the given Route. This method, upon subsequent retrievals of the Route, will return an empty set when all Detours have cleared.

### 5.65.2 Member Data Documentation

#### 5.65.2.1 String routeId [private]

The unique identifier of the affected Route.

This can be used in the TransitService method TransitService#getRoute(String) to retrieve more information about the disruption.

#### 5.65.2.2 URL transitServiceUrl [private]

The URL callback of the originating TransitService.

Clients should use this URL to obtain further distruption information, such as Detours of the affected Route.

## 5.66 RouteDisruptionEvent Class Reference

An event indicating a disruption in normal Route scheduling or service.

### Public Member Functions

- RouteDisruptionEvent (Route disruptedRoute)

  *Instantiates a new route disruption event with the affected Route.*

- Route **getDisruptedRoute** ()
- void **setDisruptedRoute** (Route disruptedRoute)

### Private Attributes

- Route disruptedRoute

  *The disrupted Route, complete with any Detour information that is available.*

### 5.66.1 Detailed Description

An event indicating a disruption in normal Route scheduling or service.

This event will be initiated by a TransitProvider in cases of mechanical failure, scheduled maintenance, infrastructure delays such as construction or road closures, etc.

Note that a RouteDisruptionEvent can signal that a Route is returning back to normal service after the disruption has cleared. This is done by sending a Route with no Detours.

### 5.66.2 Constructor & Destructor Documentation

#### 5.66.2.1 RouteDisruptionEvent ( Route *disruptedRoute* )

Instantiates a new route disruption event with the affected Route.

**Parameters**

| | |
|---|---|
| *disruptedRoute* | The Disrupted Route |

## 5.67 RouteRepository Interface Reference

A Repository Pattern supporting lifecycle operations of Routes, such as Read, Save, Delete, and Query functionality.

### Public Member Functions

- void delete (String routeId)

  *Deletes the Route corresponding to the given **routeId**.*
- void save (Route route)

  *Saves the Route to the Repository.*
- void save (Set< Route > routes)

  *Saves all of the Routes to the Repository.*
- Route read (String routeId)

  *Read a single Route from the Repository by its identifier.*
- Collection< Route > getAll ()

  *Retrieves all available Routes in the Repository.*

### 5.67.1 Detailed Description

A Repository Pattern supporting lifecycle operations of Routes, such as Read, Save, Delete, and Query functionality.

## 5.67.2 Member Function Documentation

### 5.67.2.1 void delete ( String *routeId* )

Deletes the Route corresponding to the given **routeId**.

**Precondition**

A Route with the given **routeId** exists in the Repository.

**Postcondition**

A Route with the given **routeId** is removed from the Repository and is no longer available for retrieval.

**Parameters**

| | |
|---|---|
| *routeId* | |

### 5.67.2.2 Collection<Route> getAll ( )

Retrieves all available Routes in the Repository.

**Returns**

All available Routes.

### 5.67.2.3 Route read ( String *routeId* )

Read a single Route from the Repository by its identifier.

If no Route is found with the requested **routeId**, a null value is returned.

**Parameters**

| | |
|---|---|
| *routeId* | The identifier of the requested Route |

**Returns**

The requested Route

### 5.67.2.4 void save ( Route *route* )

Saves the Route to the Repository.

**Precondition**

The Route has been validated with all appropriate business rules.

**See Also**

RouteSpecification

**Postcondition**

The Route is available for retrieval by id and also by appropriate Queries.

**Parameters**

| | |
|---|---|
| *route* | The Route to save. |

#### 5.67.2.5 void save ( Set< Route > *routes* )

Saves all of the Routes to the Repository.

**Precondition**

The Routes have been validated with all appropriate business rules.

**See Also**

RouteSpecification

**Postcondition**

The Routes are available for retrieval by id and also by appropriate Queries.

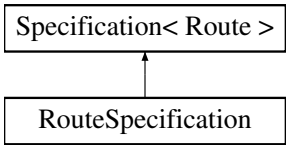
**Parameters**

| | |
|---|---|
| *route* | The Route to save. |

## 5.68 RouteSpecification Class Reference

A Specification Pattern class for validating a Route.

Inheritance diagram for RouteSpecification:




### Public Member Functions

- boolean isSatisfiedBy (Route candidate)

  *Validates the given Route with the appropriate validation rules of this Specification.*
- Specification< Route > **or** (Specification< Route > specification)
- Specification< Route > **and** (Specification< Route > specification)
- Specification< Route > **not** (Specification< Route > specification)

### 5.68.1 Detailed Description

A Specification Pattern class for validating a Route.

Validation can happen in this class, or business rules can be combined using the and, or, or not methods of separate Specifications.

This Specification is necessary because Routes aren't necessarily created by a controlled Factory, but as a result of parsing of input files via subclasses of the AbstractFeedParserTemplate. Because creation of Routes isn't centralized, it is necessary to centralize the validation rules so that subclasses of AbstractFeedParserTemplate have access to it.

This Specification must validate the following conditions:

1. Each Route has an non-null routeId
2. Each Route has an non-blank routeName
3. Each Route has two or more Stops

### 5.68.2 Member Function Documentation

#### 5.68.2.1 boolean isSatisfiedBy ( Route *candidate* )

Validates the given Route with the appropriate validation rules of this Specification.

**Parameters**

| | |
|---|---|
| *candidate* | The Route to be validated |

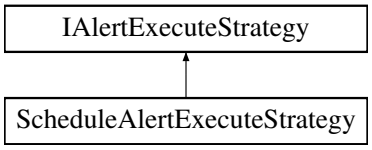
**Returns**

True if a valid Route, False if not

## 5.69 ScheduleAlertExecuteStrategy Class Reference

A concrete implementation of IAlertExecuteStrategy that handles executing alert related to user's regular schedule.

Inheritance diagram for ScheduleAlertExecuteStrategy:




### Public Member Functions

- boolean execute (Alert alertModel)

  *For each passed in alert, it pushes notification to the user informing about their upcoming trip and routes (as they have scheduled in the first place)*

### Package Attributes

- AlertRepository alertRepository

  *An instance of AlertRepository that is used to fetch alerts.*
- AlertUserClient userClient

  *A spring autowired instance of AlertUserClient that can call the User module to get user information.*

### 5.69.1 Detailed Description

A concrete implementation of IAlertExecuteStrategy that handles executing alert related to user's regular schedule.

### 5.69.2 Member Data Documentation

#### 5.69.2.1 AlertRepository alertRepository [package]

An instance of AlertRepository that is used to fetch alerts.

This is autowired via Spring Framework.

#### 5.69.2.2 AlertUserClient userClient [package]

A spring autowired instance of AlertUserClient that can call the User module to get user information.

This is autowired via Spring Framework.

## 5.70 Session Class Reference

This class represents a single session for a user of the system, and all of the state data associated with that session.

### Protected Member Functions

- String getSessionToken ()

  *This retrieves the session token.*
- Calendar getCreationTime ()

  *This retrieves the time that the session was created.*
- Calendar getExpirationTime ()

  *This retrieves the time that the session is set to expire.*
- void setExpirationTime (Calendar expirationTime)

  *This sets the time that the session should expire.*
- boolean isAlertSession ()

  *This checks to see if the session is an alert session.*
- boolean isValid ()

  *This checks to see if the session is valid.*
- void setValid (boolean valid)

  *This sets whether the session is valid.*
- int getUserId ()

  *This gets the ID of the user linked to this session.*

### Package Functions

- Session (String sessionToken, Calendar creationTime, boolean isAlertSession, int userId)

  *Create a new session object.*

### Private Attributes

- final String **sessionToken**
- final Calendar **creationTime**
- Calendar **expirationTime**
- final boolean **isAlertSession**
- boolean **valid**
- final int **userId**

## 5.70.1 Detailed Description

This class represents a single session for a user of the system, and all of the state data associated with that session.

The session should already exist in the database before instantiating this object. A session grants a user access to the data associated with that user. Sessions expire after a certain point, and can also be invalidated by a user logging out. Some sessions are designed to be longer lasting, for use with alerts. This object is not visible to clients - when they must pass a session reference, they do so by passing around the sessionToken.

## 5.70.2 Constructor & Destructor Documentation

### 5.70.2.1 Session ( String *sessionToken,* Calendar *creationTime,* boolean *isAlertSession,* int *userId* ) [package]

Create a new session object.

It is not visible to clients, as User objects should only be constructed through the UserRepository. The parameters taken by the constructor cannot be changed once the session is created.

**Parameters**

| | |
|---|---|
| *sessionToken* | unique session token |
| *creationTime* | time that the session was created |
| *isAlertSession* | true if this is an alert session, false otherwise |
| *userId* | user ID that the session is linked to |

## 5.70.3 Member Function Documentation

### 5.70.3.1 Calendar getCreationTime ( ) [protected]

This retrieves the time that the session was created.

**Returns**

session creation time

### 5.70.3.2 Calendar getExpirationTime ( ) [protected]

This retrieves the time that the session is set to expire.

**Returns**

session expiration time

### 5.70.3.3 String getSessionToken ( ) [protected]

This retrieves the session token.

**Returns**

session token

#### 5.70.3.4 int getUserId ( ) [protected]

This gets the ID of the user linked to this session.

**Returns**

user's ID number

#### 5.70.3.5 boolean isAlertSession ( ) [protected]

This checks to see if the session is an alert session.

**Returns**

true if it is, false otherwise

#### 5.70.3.6 boolean isValid ( ) [protected]

This checks to see if the session is valid.

**Returns**

true if it is, false otherwise

#### 5.70.3.7 void setExpirationTime ( Calendar *expirationTime* ) [protected]

This sets the time that the session should expire.

**Parameters**

| | |
|---|---|
| *expirationTime* | expiration time to set |

#### 5.70.3.8 void setValid ( boolean *valid* ) [protected]

This sets whether the session is valid.

**Parameters**

| | |
|---|---|
| *valid* | true if it is, false otherwise |

## 5.71 SessionRepository Class Reference

This class is responsible for handling database access for Sessions, and to construct, persist, and retrieve Session objects.

### Package Functions

- Session createSession (User user, boolean isAlertSession) throws BusBuddyInternalException

  *This creates a new session for the given user.*
- Session getSession (String sessionToken) throws BusBuddyInternalException, BusBuddyForbiddenException

  *This method gets a session from the database.*
- void killSession (String sessionToken) throws BusBuddyInternalException, BusBuddyNotFoundException

  *This method invalidates a session in the database.*

## 5.71.1 Detailed Description

This class is responsible for handling database access for Sessions, and to construct, persist, and retrieve Session objects.

## 5.71.2 Member Function Documentation

### 5.71.2.1 Session createSession ( User *user,* boolean *isAlertSession* ) throws BusBuddyInternalException [package]

This creates a new session for the given user.

#### Precondition

The User object parameter must be a valid user retrieved from the database.

#### Postcondition

A session is created in the database, and the object representing that session is returned.

#### Parameters

| | |
|---|---|
| *user* | This is the user to create the session for. |
| *isAlertSession* | This is set to true if this should be a long-lived session, for an alert. Otherwise, set to false for a normal session. |

#### Returns

The method returns the newly created Session object.

#### Exceptions

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is a database error. |

### 5.71.2.2 Session getSession ( String *sessionToken* ) throws BusBuddyInternalException, BusBuddyForbiddenException [package]

This method gets a session from the database.

In addition, since this method is only called when there is an it will update the expiration date on the session.

**Precondition**

The sessionToken parameter must be a valid session identifier in the database.

**Postcondition**

The session's expiration date will have been pushed back due to this activity in the session.

**Parameters**

| | |
|---|---|
| *sessionToken* | This is the session token that identifies the session. |

**Returns**

Session object represented by the session token that was passed in.

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is a database error. |
| *BusBuddyForbidden-Exception* | This exception is thrown if the session token is invalid or the session is expired. |

#### 5.71.2.3 void killSession ( String *sessionToken* ) throws BusBuddyInternalException, BusBuddyNotFoundException [package]

This method invalidates a session in the database.

**Precondition**

The sessionToken parameter must be a valid session identifier in the database.

**Postcondition**

The session will be invalidated and future calls using that sessionToken will fail.

**Parameters**

| | |
|---|---|
| *sessionToken* | This is the session token that identifies the session. |

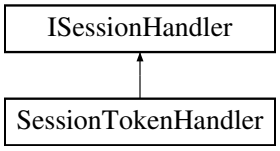
**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is a database error. |
| *BusBuddyNotFound-Exception* | This exception is thrown if the session token is invalid. |

## 5.72 SessionTokenHandler Class Reference

A concrete strategy implementation of ISessionHandler that can verify the session token being passed.

Inheritance diagram for SessionTokenHandler:




### Public Member Functions

- boolean verifySessionToken (String sessionToken) throws BusBuddyForbiddenException

  *This method takes in an encrypted session token in a form of string and then validates for its authenticity.*

### Package Attributes

- AlertUserClient userClient

  *A spring autowired instance of AlertUserClient that can call the User module to verify the session token.*

### 5.72.1 Detailed Description

A concrete strategy implementation of ISessionHandler that can verify the session token being passed.

### 5.72.2 Member Function Documentation

#### 5.72.2.1 boolean verifySessionToken ( String *sessionToken* ) throws BusBuddyForbiddenException

This method takes in an encrypted session token in a form of string and then validates for its authenticity.

**Exceptions**

| | |
|---|---|
| *BusBuddyForbidden-Exception* | • Throws an exception when the token is not valid. |

Implements ISessionHandler.

### 5.72.3 Member Data Documentation

#### 5.72.3.1 AlertUserClient userClient [package]

A spring autowired instance of AlertUserClient that can call the User module to verify the session token.

This is autowired via Spring Framework.

## 5.73 SessionVerificationFactory Class Reference

A factory that picks the appropriate strategy ISessionHandler to verify the encrypted token.

### Static Public Member Functions

- static ISessionHandler getSessionTokenVerificationStrategy (AlertInitiator alertInitiator)

  *This method takes in an AlertInitiator and depending upon the value can invoke different strategy to verify the token.*

### 5.73.1 Detailed Description

A factory that picks the appropriate strategy ISessionHandler to verify the encrypted token.

### 5.73.2 Member Function Documentation

#### 5.73.2.1 static ISessionHandler getSessionTokenVerificationStrategy ( AlertInitiator *alertInitiator* ) [static]

This method takes in an AlertInitiator and depending upon the value can invoke different strategy to verify the token.

**Parameters**

| | |
|---|---|
| *alertInitiator* | A valid AlertInitiator |

**Returns**

A implementation of ISessionHandler

## 5.74 Specification< T > Interface Reference

A Generic Specification to be used for chaining business validation rules together.

### Public Member Functions

- boolean isSatisfiedBy (T candidate)

  *Checks if the given candidate satisfies the specification.*
- Specification< T > or (Specification< T > specification)

  *Returns a Specification representing the 'OR' boolean operation of the Specifications.*
- Specification< T > and (Specification< T > specification)

  *Returns a Specification representing the 'AND' boolean operation of the Specifications.*
- Specification< T > not (Specification< T > specification)

  *Returns a Specification representing the 'NOT' boolean operation of the Specifications.*

### 5.74.1 Detailed Description

A Generic Specification to be used for chaining business validation rules together.

**Parameters**

| | |
|---|---|
| *<T>* | The candidate Type accepted by the Specification. |

### 5.74.2 Member Function Documentation

#### 5.74.2.1 Specification<T> and ( Specification< T > *specification* )

Returns a Specification representing the 'AND' boolean operation of the Specifications.

**Parameters**

| | |
|---|---|
| *specification* | The Specification to apply the 'AND' operation to. |

**Returns**

The 'AND' Specification

#### 5.74.2.2 boolean isSatisfiedBy ( T *candidate* )

Checks if the given candidate satisfies the specification.

**Parameters**

| | |
|---|---|
| *candidate* | The candidate |

**Returns**

true, if is satisfied by the candidate

#### 5.74.2.3 Specification<T> not ( Specification< T > *specification* )

Returns a Specification representing the 'NOT' boolean operation of the Specifications.

**Parameters**

| | |
|---|---|
| *specification* | The Specification to apply the 'NOT' operation to. |

**Returns**

The 'NOT' Specification

#### 5.74.2.4 Specification<T> or ( Specification< T > *specification* )

Returns a Specification representing the 'OR' boolean operation of the Specifications.

**Parameters**

| | |
|---|---|
| *specification* | The Specification to apply the 'OR' operation to. |

**Returns**

The 'OR' Specification

## 5.75 Stop Class Reference

A point on a Route in which a TransitVehicle will stop to pick up and drop off passengers.

### Public Member Functions

- Set< Date > getStopTimes (Date begin, Date end)

  *Reports the expected times in which a TransitVehicle will be at the given Stop for a given time period.*
- Location **getLocation** ()
- void **setLocation** (Location location)
- String **getDescription** ()
- void **setDescription** (String description)

- String **getStopId** ()
- void **setStopId** (String stopId)
- boolean **isCovered** ()
- void **setCovered** (boolean covered)

### Protected Member Functions

- Stop (String stopId, String description, Location location)

  *Instantiates a new Stop with all the mandatory fields.*

### Private Attributes

- String stopId

  *A unique identifier for this Stop.*
- String description

  *A short text-based description of describing the Stop and its location.*
- Location location

  *The physical location of the Stop.*
- boolean covered

  *Whether or not this Stop is protected by some sort of enclosure.*

## 5.75.1 Detailed Description

A point on a Route in which a TransitVehicle will stop to pick up and drop off passengers.

A Stop also is responsible for providing a set of the times in which the TransitVehicle will be at the Stop.

A Stop is identified within the context of a single Route. This means that two Routes may share the same physical Stop location, but maintain different schedules.

## 5.75.2 Constructor & Destructor Documentation

### 5.75.2.1 Stop ( String *stopId,* String *description,* Location *location* ) [protected]

Instantiates a new Stop with all the mandatory fields.

**Parameters**

| | |
|---|---|
| *stopId* | The unique identifier |
| *description* | The description |
| *location* | The Location |

## 5.75.3 Member Function Documentation

### 5.75.3.1 Set<Date> getStopTimes ( Date *begin,* Date *end* )

Reports the expected times in which a TransitVehicle will be at the given Stop for a given time period.

**Precondition**

**begin** $<$ **end**.

**Parameters**

| | |
|---|---|
| *begin* | The start of the reporting time period. All Stop Times returned will be on (or after) this time. If null, assume to be the current time. |
| *end* | The end of the reporting time period. All Stop Times returned will before this time. |

**Returns**

Stop Times associated with this Stop that satisfy the begin and end criteria.

### 5.75.4 Member Data Documentation

#### 5.75.4.1 boolean covered [private]

Whether or not this Stop is protected by some sort of enclosure.

The type of enclosure required to constitute 'covered' defined by each TransitProvider, but at a minimum there should be a roof protecting the immediate area before riders board the vehicle.

#### 5.75.4.2 String description [private]

A short text-based description of describing the Stop and its location.

This could be an intersection:

- "18th Ave and 58th St." or a landmark/park/attraction:

"Como Zoo North Entrance" or other identifying text.

This description should be sufficient enough to allow a user to to find the given Stop without necessarily needing the Location information.

#### 5.75.4.3 String stopId [private]

A unique identifier for this Stop.

In some cases, it could be a String representation of the location - or possibly the description.

As a Stop is part of the Route Aggregate, this identifier needs only be unique within the context of a Route.

## 5.76 TrackingAlertFactory Class Reference

The Alert Factory handles the creation of a user alert.

### Public Member Functions

- TrackingAlertObserver createAlertObserver (TransitVehicle vehicle)

### 5.76.1 Detailed Description

The Alert Factory handles the creation of a user alert.

The necessary values for an alert will be entered by a registered user from the BusBuddy User Interface. See UserTrackingAlertObject for input parameter details.

**Postcondition**

New user alert registered with a transit vehicle in the array list.

### 5.76.2 Member Function Documentation

#### 5.76.2.1 TrackingAlertObserver createAlertObserver ( TransitVehicle *vehicle* )

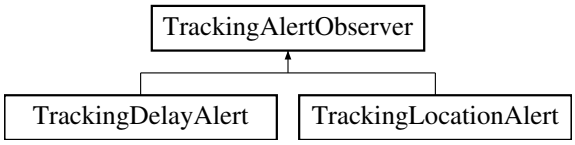
1. Determine what type of tracking alert observer to create

1. Determine what rules are needed and add an Alert Specification to this alert. Configuration for alert logic will be obtained based on a configuration file.

## 5.77 TrackingAlertObserver Class Reference

Abstract class defining the methods for the tracking alert observer.

Inheritance diagram for TrackingAlertObserver:




### Package Functions

- AlertSpecification getSpec ()

  *Return the specification to use to determine if a vehicle is in an alert range.*
- void setSpec (AlertSpecification spec)

  *Set the alert specification.*
- abstract void updateAlert ()

  *The observer pattern update method called from the subject TransitVehicle when a vehicle is determined to be in an alert range and a user needs to be notified.*

### Private Attributes

- UserTrackingAlertObject userAlertTrackingObject

  *Value Object containing the items necessary for an alert.*
- AlertSpecification specification

  *The business logic specification of how to determine if an alert needs to be sent for a vehicle.*
- AlertRequestController arc = new AlertRequestController()

  *Alert Module Controller called via a REST API processUserAlertRequest()*

### 5.77.1 Detailed Description

Abstract class defining the methods for the tracking alert observer.

**Precondition**

Vehicle is determined by AlertSpecification to within range

**Postcondition**

Alert request is sent to the AlertService Abstract class defining the methods for the tracking alert observer. This class calls alert module's alert controller via REST call to fetch necessary information.

### 5.77.2 Member Function Documentation

#### 5.77.2.1 AlertSpecification getSpec ( ) [package]

Return the specification to use to determine if a vehicle is in an alert range.

**Return values**

| | |
|---|---|
| *AlertSpecification* | |

#### 5.77.2.2 void setSpec ( AlertSpecification *spec* ) [package]

Set the alert specification.

**Parameters**

| | |
|---|---|
| *spec* | AlertSpecification - the rules used by the subject to determine if an alert is necessary. |

#### 5.77.2.3 abstract void updateAlert ( ) [package], [pure virtual]

The observer pattern update method called from the subject TransitVehicle when a vehicle is determined to be in an alert range and a user needs to be notified.

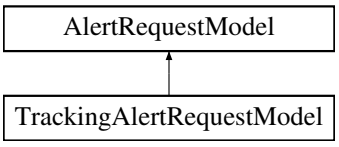
Alert notifications are actually sent using the IAlertService

Implemented in TrackingLocationAlert, and TrackingDelayAlert.

## 5.78 TrackingAlertRequestModel Class Reference

This model is a JSON representation of the request from Tracking module.

Inheritance diagram for TrackingAlertRequestModel:




### Additional Inherited Members

### 5.78.1 Detailed Description

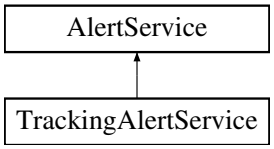
This model is a JSON representation of the request from Tracking module.

During implementation, any additional data that is needed can be added.

## 5.79 TrackingAlertService Class Reference

Extends AlertService and provides CRUD methods to operate on alerts as well as sends alerts to execute via appropriate strategy.

Inheritance diagram for TrackingAlertService:




### Public Member Functions

- AlertResponseModel createAlert (AlertRequestModel requestModel)

  *{*
- AlertResponseModel updateAlert (Alert alertModel)

  *{*
- boolean sendAlert ()

  *{*

### Additional Inherited Members

### 5.79.1 Detailed Description

Extends AlertService and provides CRUD methods to operate on alerts as well as sends alerts to execute via appropriate strategy.

This handles all the alerts initiated by Tracking module.

### 5.79.2 Member Function Documentation

#### 5.79.2.1 AlertResponseModel createAlert ( AlertRequestModel *requestModel* ) [virtual]

{

**See Also**

AlertService::createAlert(AlertRequestModel)}

Implements AlertService.

#### 5.79.2.2 boolean sendAlert ( ) [virtual]

{

**See Also**

AlertService::sendAlert()}

Implements AlertService.

#### 5.79.2.3 AlertResponseModel updateAlert ( Alert *alertModel* ) [virtual]

{

**See Also**

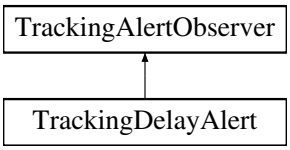
AlertService::updateAlert(Alert)}

Implements AlertService.

## 5.80 TrackingDelayAlert Class Reference

Tracking Alert Observer implements the abstract tracking alert observer and provides the method to actually send an alert to a registered user that their bus is approaching their stop.

Inheritance diagram for TrackingDelayAlert:




### Public Member Functions

- void updateAlert ()

  *Receives the notification indicating that a vehicle is in the alert range.*

### Additional Inherited Members

### 5.80.1 Detailed Description

Tracking Alert Observer implements the abstract tracking alert observer and provides the method to actually send an alert to a registered user that their bus is approaching their stop.

**Precondition**

Vehicle is determined by AlertSpecification to within range

**Postcondition**

Alert request is sent to the AlertService

### 5.80.2 Member Function Documentation

#### 5.80.2.1 void updateAlert ( ) [virtual]

Receives the notification indicating that a vehicle is in the alert range.

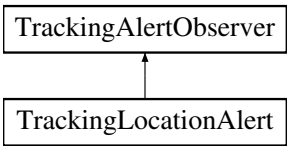
Use the AlertService to contact the registered user.

Implements TrackingAlertObserver.

## 5.81 TrackingLocationAlert Class Reference

Concrete implementation of the tracking alert observer.

Inheritance diagram for TrackingLocationAlert:




### Public Member Functions

- TrackingLocationAlert (TransitVehicle vehicle)

  *Tracking Location Alert constructor.*
- void updateAlert ()

  *Vehicle is in vicinity where user registered to be notified, calls Alert Service.*

### Additional Inherited Members

#### 5.81.1 Detailed Description

Concrete implementation of the tracking alert observer.

The subject calls the update alert for registered vehicles.

#### 5.81.2 Constructor & Destructor Documentation

##### 5.81.2.1 TrackingLocationAlert ( TransitVehicle *vehicle* )

Tracking Location Alert constructor.

Associates user information with vehicle to monitor.

**Parameters**

| | |
|---|---|
| *vehicle* | TransitVehicle Vehicle to add alert monitoring |
| *alert* | - TrackingLocationAlert User contact details and vehicle location indicating when user wants to receive alert. |

## 5.82 TrackingResponseModel Class Reference

This is a basic tracking response model that is returned for every tracking related request.

### Public Member Functions

- String formatJSONResponse ()

  *Formats output responses and requests from the tracking module.*
- VehicleObject convertJSONVehicleInput ()

  *Converts input JSON formatted vehicle registration request to internal vehicle object.*
- UserTrackingAlertObject convertJSONAlertInput ()

  *Convert input JSON formatted alert request from user interface to internal user alert object.*

### Private Attributes

- String status

  *Status message for the tracking request.*
- String errorMessage

  *Any error message if the tracking request fails.*

### 5.82.1 Detailed Description

This is a basic tracking response model that is returned for every tracking related request.

Additional parameters can be added as needed during implementation phase. This model is responsible for verify that all data received is within range before returning local object.

### 5.82.2 Member Function Documentation

#### 5.82.2.1 UserTrackingAlertObject convertJSONAlertInput ( )

Convert input JSON formatted alert request from user interface to internal user alert object.

**Returns**

UserTrackingAlertObject values

#### 5.82.2.2 VehicleObject convertJSONVehicleInput ( )

Converts input JSON formatted vehicle registration request to internal vehicle object.

**Returns**

VehicleObject created from user registration.

#### 5.82.2.3 String formatJSONResponse ( )

Formats output responses and requests from the tracking module.

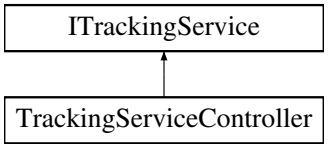
**Returns**

String containing response

## 5.83 TrackingServiceController Class Reference

Tracking service controller is the concrete implementation of the tracking service interface.

Inheritance diagram for TrackingServiceController:



### Public Member Functions

- void registerVehicleOnRoute (URL url, int gpsDeviceID)

  *Create a vehicle when a user registers a vehicle on a route through the user interface.*
- void unregisterVehicleFromRoute (String url, int gpsDeviceID)

  *Remove a vehicle from the vehicle repository when the vehicle is no longer in service.*
- void addUserTrackingAlert (UserTrackingAlertObject utao)

  *Add a new user alert.*
- void startTrackingController ()

  *Gets a list of saved alerts from the AlertService and restores then on tracking module startup.*
- GPSLocationObject getTransitVehicleLocation (int gpsDeviceID)

  *Find locations of the specified GPS device ID.*

### Package Attributes

- TransitVehicleFactory transitFactory = new TransitVehicleFactory()

  *Logic for creating transit vehicles.*
- TrackingAlertFactory alertFactory = new TrackingAlertFactory()

  *Logic for creating new user alerts.*

## 5.83.1 Detailed Description

Tracking service controller is the concrete implementation of the tracking service interface.

Provides the tracking functionality to other Bus Buddy modules and ties vehicle location to registered user alerts. The tracking service controller accepts REST requests from the User Interface module, the Alert Module

## 5.83.2 Member Function Documentation

### 5.83.2.1 void addUserTrackingAlert ( UserTrackingAlertObject *utao* )

Add a new user alert.

Necessary inputs are entered by the user on the User Interface and made available to the Tracking Controller through UserTrackingAlertObject

1. Verify that there is a vehicle registered on the routed requested by the user.
2. Get a list of vehicles on the route from the vehicle repository
3. Create a new Tracking Alert Observer
4. Add an alert specification containing the business rules to determine if bus is in alert range.
5. Register the user alert observer to the vehicles

Find the vehicles registered on this route

Create an alert for this user request and register this alert with the vehicle(s) the user is watching.

Implements ITrackingService.

### 5.83.2.2 void registerVehicleOnRoute ( URL *url*, int *gpsDeviceID* )

Create a vehicle when a user registers a vehicle on a route through the user interface.

Accepts a VehicleObject JSON

Implements ITrackingService.

#### 5.83.2.3 void unregisterVehicleFromRoute ( String *url*, int *gpsDeviceID* )

Remove a vehicle from the vehicle repository when the vehicle is no longer in service.

**Parameters**

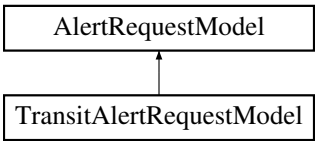
| | |
|---|---|
| *url* | - URL transit company and route information for this vehicle |
| *gpsDeviceID* | - integer GPS device ID installed in vehicle, must match the ID the vehicle registered with. |

Implements ITrackingService.

## 5.84 TransitAlertRequestModel Class Reference

This model is a JSON representation of the request from Transit module.

Inheritance diagram for TransitAlertRequestModel:




### Additional Inherited Members

### 5.84.1 Detailed Description

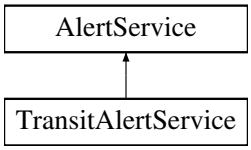
This model is a JSON representation of the request from Transit module.

During implementation, any additional data that is needed can be added.

## 5.85 TransitAlertService Class Reference

Extends AlertService and provides CRUD methods to operate on alerts as well as sends alerts to execute via appropriate strategy.

Inheritance diagram for TransitAlertService:




### Public Member Functions

- AlertResponseModel createAlert (AlertRequestModel requestModel)
  *{*
- AlertResponseModel updateAlert (Alert alertModel)
  *{*
- boolean sendAlert ()
  *{*

### Additional Inherited Members

### 5.85.1 Detailed Description

Extends AlertService and provides CRUD methods to operate on alerts as well as sends alerts to execute via appropriate strategy.

This handles all the alerts initiated by Transit module.

### 5.85.2 Member Function Documentation

#### 5.85.2.1 AlertResponseModel createAlert ( AlertRequestModel *requestModel* ) [virtual]

{

**See Also**

AlertService::createAlert(AlertRequestModel)}

Implements AlertService.

#### 5.85.2.2 boolean sendAlert ( ) [virtual]

{

**See Also**

AlertService::sendAlert()}

Implements AlertService.

#### 5.85.2.3 AlertResponseModel updateAlert ( Alert *alertModel* ) [virtual]

{

**See Also**

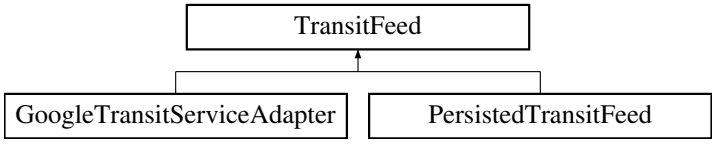
AlertService::updateAlert(Alert)}

Implements AlertService.

## 5.86 TransitFeed Interface Reference

A TransitFeed is an abstraction over a service or set of services that provide information about Routes.

Inheritance diagram for TransitFeed:



### Public Member Functions

- Route getRoute (String routeId)

  *Gets a Route by its unique identifier.*
- Set< Route > getRoutes (Location pickup, Location dropoff, int distance)

  *Gets all available Routes that match a **pickup** or **dropoff** Location by not more than a given **distance**.*

## 5.86.1 Detailed Description

A TransitFeed is an abstraction over a service or set of services that provide information about Routes.

This differs from the TransitService interface in that a TransitFeed does not expose provenance information such as the TransitService#getTransitInfo method.

Because of this, a single TransitService (or TransitProvider) could use more than one TransitFeed to expose their Transit data. For example, a TransitProvider could utilize two different database storage schemes, each one represented as a separate TransitFeed, and then aggregate the two into one TransitService.

## 5.86.2 Member Function Documentation

### 5.86.2.1 Route getRoute ( String *routeId* )

Gets a Route by its unique identifier.

#### Precondition

**routeId** is not null or blank.

#### Postcondition

The Route is returned if the **routeId** is found, else null.

#### Parameters

| | |
|---|---|
| *routeId* | The unique identifier of the Route |

#### Returns

The matching Route, or null if not found

Implemented in GoogleTransitServiceAdapter, and PersistedTransitFeed.

### 5.86.2.2 Set<Route> getRoutes ( Location *pickup*, Location *dropoff*, int *distance* )

Gets all available Routes that match a **pickup** or **dropoff** Location by not more than a given **distance**.

#### Precondition

**pickup** is not null or blank.
**dropoff** is not null or blank.
**distance** is non-negative.

**Parameters**

| | |
|---|---|
| *pickup* | The requested pickup Location |
| *pickup* | The requested dropoff Location |
| *distance* | The distance (in miles) that each Route can deviate from the requested **pickup** or **dropoff** Location. For each Route returned, neither its start or end Location can differ from the requested **pickup** or **dropoff** Location by more than the value of the **distance** parameter. |

**Returns**

The matching Routes

Implemented in GoogleTransitServiceAdapter, and PersistedTransitFeed.

## 5.87 TransitInfo Class Reference

An immutable Value Object describing metadata about a TransitService.

### Public Member Functions

- String **getTransitAuthorityName** ()
- void **setTransitAuthorityName** (String transitAuthorityName)
- URL **getWebsite** ()
- void **setWebsite** (URL website)
- byte[] **getLogo** ()
- void **setLogo** (byte[] logo)

### Private Attributes

- String transitAuthorityName

  *The name of the governing body of the associated TransitService.*
- URL website

  *A resolvable URL reference to the official Transit Authority web site.*
- byte[] logo

  *A byte array of the Transit Authority logo, if any.*

### 5.87.1 Detailed Description

An immutable Value Object describing metadata about a TransitService.

Each TransitService is required to supply the following information.

### 5.87.2 Member Data Documentation

#### 5.87.2.1 byte [] logo [private]

A byte array of the Transit Authority logo, if any.

Allowed formats are unspecified, as image format parsing/conversion is beyond the scope of this project.

#### 5.87.2.2 String transitAuthorityName [private]

The name of the governing body of the associated TransitService.

This can be a Federal, State, or Local governing body responsible for the transit activity associated with the TransitService.

#### 5.87.2.3 URL website [private]

A resolvable URL reference to the official Transit Authority web site.

Where possible, this site should contain contact info and links to policy, specialized transit requirements, or other information.

## 5.88 TransitProvider Class Reference

A TransitProvider is a description of a company or organization that is the producer of public transportation services.

### Private Attributes

- String providerId

  *A unique identifier that globally identifies this TransitProvider.*

- Set< TransitProviderObserver > transitProviderObserver

  *An Observer Pattern mechanism to alertlink TransitProviderObserver}s of changes in Route service;.*
- String name

  *A text description of the TransitProvider.*
- abstract void fireRouteDistruptionEvent ()

  *This method is called internally by the TransitProvider to fire a RouteDisruptionEvent.*
- String **getProviderId** ()
- void **setProviderId** (String providerId)
- String **getName** ()
- void **setName** (String name)
- void registerObserver (TransitProviderObserver transitProviderObserver)

  *Register the provided TransitProviderObserver as an observer to this class.*
- void unregisterObserver (TransitProviderObserver transitProviderObserver)

  *Unregister the provided TransitProviderObserver as an observer to this class.*

### 5.88.1 Detailed Description

A TransitProvider is a description of a company or organization that is the producer of public transportation services.

Observers may subscribe to a TransitProvider to receive updates on Route disruptions, such as changes in service availability or schedule.

### 5.88.2 Member Function Documentation

#### 5.88.2.1 abstract void fireRouteDistruptionEvent ( ) [protected], [pure virtual]

This method is called internally by the TransitProvider to fire a RouteDisruptionEvent.

TransitProvider subclasses will determine when these Events are fired – for example, if a TransitProvider has scheduled maintenance days, or known outages due to mechanical breakdown.

After a RouteDisruptionEvent is fired, this class will perform the following:

- Notify all TransitProviderObservers of the disruption with the updated Route. This updated Route should include all necessary Detour information.

#### 5.88.2.2 void registerObserver ( TransitProviderObserver *transitProviderObserver* )

Register the provided TransitProviderObserver as an observer to this class.

**Parameters**

| | |
|---|---|
| *transitProvider-Observer* | The TransitProviderObserver to unregister. |

#### 5.88.2.3 void unregisterObserver ( TransitProviderObserver *transitProviderObserver* )

Unregister the provided TransitProviderObserver as an observer to this class.

**Parameters**

| | |
|---|---|
| *transitProvider-Observer* | The TransitProviderObserver to unregister. |

### 5.88.3 Member Data Documentation

#### 5.88.3.1 String name [private]

A text description of the TransitProvider.

This is the text that will be displayed on guides, Route maps, and advertisements.

#### 5.88.3.2 String providerId [private]

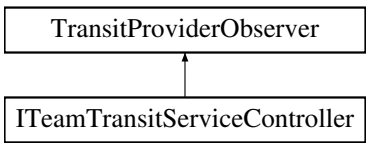
A unique identifier that globally identifies this TransitProvider.

The actual identifier type is unspecified and left to implementations. It may be the same as the name, if that is sufficient to provide uniqueness.

## 5.89 TransitProviderObserver Interface Reference

An asynchronous update interface for receiving notifications about TransitProvider Route disruptions.

Inheritance diagram for TransitProviderObserver:




### Public Member Functions

- void handleRouteDisruptionEvent (RouteDisruptionEvent routeDisruptionEvent)

  *This method is called when a TransitProvider fires a RouteDisruptionEvent indicating a disruption in normal services and/or schedules.*

### 5.89.1 Detailed Description

An asynchronous update interface for receiving notifications about TransitProvider Route disruptions.

### 5.89.2 Member Function Documentation

#### 5.89.2.1 void handleRouteDisruptionEvent ( RouteDisruptionEvent *routeDisruptionEvent* )

This method is called when a TransitProvider fires a RouteDisruptionEvent indicating a disruption in normal services and/or schedules.

**Parameters**

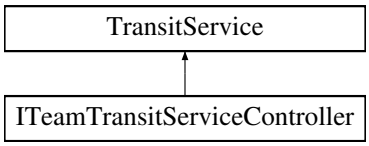
| | |
|---|---|
| *routeDisruption-Event* | The Route Disruption Event |

Implemented in ITeamTransitServiceController.

## 5.90 TransitService Interface Reference

The TransitService is an interface to get Route/Fare/Detour information from a TransitProvider.

Inheritance diagram for TransitService:




### Public Member Functions

- Route getRoute (String routeId) throws BusBuddyNotFoundException

  *Gets a Route by its unique identifier.*
- Set< Route > getRoutes (Location pickup, Location dropoff, int distance)

  *Gets all available Routes that match a **pickup** or **dropoff** Location by not more than a given **distance**.*
- TransitInfo getTransitInfo ()

  *Gets metadata about the Transit Authority providing the information retrieved from this service.*
- URL getServiceURL ()

  *The URL that uniquely identifies this TransitService.*

### 5.90.1 Detailed Description

The TransitService is an interface to get Route/Fare/Detour information from a TransitProvider.

This service will provide a consistent interface for the application logic to query to get this information.

### 5.90.2 Member Function Documentation

#### 5.90.2.1 Route getRoute ( String *routeId* ) throws BusBuddyNotFoundException

Gets a Route by its unique identifier.

**Parameters**

| | |
|---|---|
| *routeId* | The unique identifier of the Route |

**Returns**

The matching Route

**Precondition**

**routeId** is not null or blank.

**Postcondition**

The Route is returned if the **routeId** is found, else null.

**Exceptions**

| | |
|---|---|
| *BusBuddyNotFound-Exception* | Thrown if the Route with the given **routeId** is not found. |

Implemented in ITeamTransitServiceController.

#### 5.90.2.2 Set<Route> getRoutes ( Location *pickup,* Location *dropoff,* int *distance* )

Gets all available Routes that match a **pickup** or **dropoff** Location by not more than a given **distance**.

**Parameters**

| | |
|---|---|
| *pickup* | The requested dropoff Location |
| *dropoff* | the dropoff |
| *distance* | The distance (in miles) that each Route can deviate from the requested **pickup** or **dropoff** |

**Returns**

The matching Routes

**Precondition**

**pickup** is not null or blank.
**dropoff** is not null or blank.
**distance** is non-negative. Location. For each Route returned, neither its start or end Location can differ from the requested **pickup** or **dropoff** Location by more than the value of the **distance** parameter.

Implemented in ITeamTransitServiceController.

#### 5.90.2.3 URL getServiceURL ( )

The URL that uniquely identifies this TransitService.

In a REST environment, this might be the root of the REST API path.

**Returns**

The URL of this service

Implemented in ITeamTransitServiceController.

#### 5.90.2.4 TransitInfo getTransitInfo ( )

Gets metadata about the Transit Authority providing the information retrieved from this service.

**Returns**

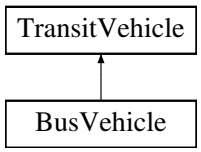
The TransitInfo of the Transit Authority of this service.

Implemented in ITeamTransitServiceController.

## 5.91 TransitVehicle Class Reference

Abstract transit vehicle class contains the common data for all types of vehicles and the Subject GPS Tracking and the GPS observer to receive GPS location updates.

Inheritance diagram for TransitVehicle:




### Public Member Functions

- abstract void registerTrackingAlert (TrackingAlertObserver ao)

  *Transit Vehicle is also the subject for tracking user subscribed alerts.*
- abstract void **unregisterTrackingAlert** (TrackingAlertObserver ao)
- void checkForAlerts ()

  *The Observer Pattern Subject notify method extended to limit the number of alerts issued by check conditions prior to actually triggering an alert.*
- void triggerAlert (TrackingAlertObserver ao)

  *The conditions in the Alert Specification were met, send update to the observer.*
- void addAlertSpecification ()

  *Add an alert specification AlertSpecification to this vehicle.*
- void removeAlertSpecifcation ()

  *Remove an alert specification from a transit vehicle.*
- String toString ()

  *Provide a generic method to output Transit Vehicle information.*

### Private Attributes

- VehicleObject vehicle

  *Value Object holding vehicle details.*
- GPSLocationObserver gpsObserver

  *Observer that update the GPS coordinates of the vehicle as they are received.*
- ArrayList< AlertSpecification > alertSpecification

  *Rules to determine if this vehicle is in an alert range.*

### 5.91.1 Detailed Description

Abstract transit vehicle class contains the common data for all types of vehicles and the Subject GPS Tracking and the GPS observer to receive GPS location updates.

### 5.91.2 Member Function Documentation

#### 5.91.2.1 void addAlertSpecification ( )

Add an alert specification AlertSpecification to this vehicle.

A vehicle may have these alerts:

1. one or more users registered for location based alerts
2. transit company registered for delay alerts, or loss of GPS signal alerts

#### 5.91.2.2 void checkForAlerts ( )

The Observer Pattern Subject notify method extended to limit the number of alerts issued by check conditions prior to actually triggering an alert.

The checkForAlerts method uses AlertSpecification to determine if the observing vehicle should be notified.

## 5.92 TransitVehicleFactory Class Reference

Transit Vehicle Factory encapsulates the complexity of creating a new vehicle.

### Public Member Functions

- TransitVehicle createTransitVehicle (URL url, int gpsDeviceID)

### Protected Member Functions

- int getVehicleGPSDeviceID (URL url)

  *Retrieve the GPS Device ID from repository of vehicles registered for route identified by type and URL.*

### Private Member Functions

- int getGPSTypeFromURL (URL url)

  *Parse the input URL for information about connecting to GPS device in this vehicle.*

### 5.92.1 Detailed Description

Transit Vehicle Factory encapsulates the complexity of creating a new vehicle.

Inputs are obtained from the user interface when a vehicle is registered by a user.

### 5.92.2 Member Function Documentation

#### 5.92.2.1 TransitVehicle createTransitVehicle ( URL *url*, int *gpsDeviceID* )

Determine what type of vehicle is needed.

Determine what type of GPS tracking is available on this vehicle and register with the appropriate GPSLocation-Tracking service.

#### 5.92.2.2 int getGPSTypeFromURL ( URL *url* ) [private]

Parse the input URL for information about connecting to GPS device in this vehicle.

**Parameters**

| | |
|---|---|
| *url* | - URL from User Interface, contains GPS connection information. |

**Returns**

integer type of GPS Device Commercial Service, GPS Pusher, or GPS Puller.

#### 5.92.2.3 int getVehicleGPSDeviceID ( URL *url* ) [protected]

Retrieve the GPS Device ID from repository of vehicles registered for route identified by type and URL.

**Parameters**

| | |
|---|---|
| *url* | - URL identifying the transit company |

**Returns**

integer GPS Device ID

## 5.93 Trip Class Reference

A Trip is considered an ordered collection of Routes going from a starting point to an ending point.

### Public Member Functions

- List< Route > **getRoutes** ()
- void **setRoutes** (List< Route > routes)

### Protected Member Functions

- Trip (List< Route > routes)

  *Instantiates a new Trip with the appropriate Routes.*

### Private Attributes

- List< Route > routes

  *The ordered collection of Routes that when combined make a navigable Trip.*

### 5.93.1 Detailed Description

A Trip is considered an ordered collection of Routes going from a starting point to an ending point.

A Trip can be thought of as a composition of Routes, and the TripService is the service that composes them.

**Invariant**

routes.size() > 0

### 5.93.2 Constructor & Destructor Documentation

#### 5.93.2.1 Trip ( List< Route > *routes* ) [protected]

Instantiates a new Trip with the appropriate Routes.

**Precondition**

**routes**.size() > 0

**Parameters**

| | |
|---|---|
| *routes* | The Routes of this Trip |

## 5.94 TripInformation Class Reference

This model stores the information about a trip as a value object.

### Public Member Functions

- List< String > getRouteIds ()

  *Provides a list of routeId in the Trip model.*
- Trip **getTripData** ()
- void **setTripData** (Trip tripData)
- Date **getLastModifiedDate** ()
- void **setLastModifiedDate** (Date lastModifiedDate)
- Date **getCreatedDate** ()
- void **setCreatedDate** (Date createdDate)

### Private Attributes

- Trip tripData

  *Necessary data about a trip.*
- Date lastModifiedDate

  *Last date when the trip Information was modified or updated.*
- Date createdDate

  *Date when the actual trip was created.*

### 5.94.1 Detailed Description

This model stores the information about a trip as a value object.

Hence, we have created and last modified date to track the freshness of data. Currently, the Trip is referring to the model in Trip module. But we expect this to be stored as a value object and during implementation we can create a copy of its model for alert module.

### 5.94.2 Member Function Documentation

#### 5.94.2.1 List<String> getRouteIds ( )

Provides a list of routeId in the Trip model.

**Returns**

### 5.94.3 Member Data Documentation

#### 5.94.3.1 Trip tripData [private]

Necessary data about a trip.

This contains a collection of routes. {

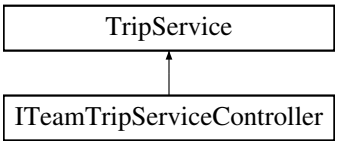
**See Also**

Trip}

## 5.95 TripService Interface Reference

A Service to calculate a collection of Routes, or a Trip, allowing for a continuous transit path from a start Location to an end Location.

Inheritance diagram for TripService:




### Public Member Functions

- Trip calculateTrip (Location start, Location end)

  *Calculate an optimal Trip given a **start** Location and an **end** Location.*

### 5.95.1 Detailed Description

A Service to calculate a collection of Routes, or a Trip, allowing for a continuous transit path from a start Location to an end Location.

### 5.95.2 Member Function Documentation

#### 5.95.2.1 Trip calculateTrip ( Location *start,* Location *end* )

Calculate an optimal Trip given a **start** Location and an **end** Location.

**Parameters**

| | |
|---|---|
| *start* | The requested start Location of the Trip. |
| *end* | The requested end Location of the Trip. |

**Returns**

The calculated Trip

Implemented in ITeamTripServiceController.

## 5.96 User Class Reference

This class represents a single user of the system, and all of the state data associated with that user.

### Public Member Functions

- int getUserId ()

  *This retrieves the user identifier.*
- String getUsername ()

  *This retrieves the user's username.*
- boolean isForcePasswordChange ()

  *This checks to see if the password is in a state where it needs to be changed.*
- String getFirstName ()

  *This retrieves the user's first name.*
- void setFirstName (String firstName)

  *This sets the user's first name.*
- String getEmail ()

  *This retrieves the user's e-mail.*
- void setEmail (String email)

  *This sets the user's e-mail.*
- Short getCountryCode ()

  *This gets the user's country code.*
- void setCountryCode (Short countryCode)

  *This sets the user's country code.*
- String getMobile ()

  *This gets the user's mobile phone number.*
- void setMobile (String mobile)

  *This sets the user's mobile phone number.*
- UserType getUserType ()

  *This retrieves the type of the current user.*
- void setUserType (UserType userType)

  *This sets the type of the current user.*

### Protected Member Functions

- User (int userId, String username)

  *This constructs a new User object.*
- String getPasswordHash ()

  *This retrieves the password hash for this user.*
- void setPasswordHash (String passwordHash)

  *This sets the password hash for this user.*
- void setForcePasswordChange (boolean forcePasswordChange)

  *This sets the state indicating if the password is in a state where it needs to be changed.*

### Private Attributes

- final int **userId**
- final String **username**
- String **password**
- String **passwordHash**
- boolean **forcePasswordChange**
- String **firstName**
- String **email**
- Short **countryCode**
- String **mobile**
- UserType **userType**

## 5.96.1 Detailed Description

This class represents a single user of the system, and all of the state data associated with that user.

The user should already exist in the database before instantiating this object.

## 5.96.2 Constructor & Destructor Documentation

### 5.96.2.1 User ( int *userId,* String *username* ) [protected]

This constructs a new User object.

It is not visible to clients, as User objects should only be constructed through the UserRepository.

**Parameters**

| | |
|---|---|
| *userId* | This is the user's unique identifier, which should match the database. |
| *username* | This is the user's username. It cannot be changed. |

## 5.96.3 Member Function Documentation

### 5.96.3.1 Short getCountryCode ( )

This gets the user's country code.

**Returns**

user's country code

### 5.96.3.2 String getEmail ( )

This retrieves the user's e-mail.

**Returns**

user's e-mail

### 5.96.3.3 String getFirstName ( )

This retrieves the user's first name.

**Returns**

user's first name

#### 5.96.3.4 String getMobile ( )

This gets the user's mobile phone number.

**Postcondition**

The mobile phone number returned should be a String containing only digits.

**Returns**

user's mobile phone number

#### 5.96.3.5 String getPasswordHash ( ) [protected]

This retrieves the password hash for this user.

It has decreased visibility and is ignored when serializing responses, as this data should not be shared beyond this module.

**Returns**

hash of the user's password

#### 5.96.3.6 int getUserId ( )

This retrieves the user identifier.

**Returns**

user identifier

#### 5.96.3.7 String getUsername ( )

This retrieves the user's username.

**Returns**

username

#### 5.96.3.8 UserType getUserType ( )

This retrieves the type of the current user.

**Returns**

user type

#### 5.96.3.9 boolean isForcePasswordChange ( )

This checks to see if the password is in a state where it needs to be changed.

**Returns**

true if it is, false if it is not

#### 5.96.3.10 void setCountryCode ( Short *countryCode* )

This sets the user's country code.

**Parameters**

| | |
|---|---|
| *countryCode* | user's country code |

#### 5.96.3.11 void setEmail ( String *email* )

This sets the user's e-mail.

**Parameters**

| | |
|---|---|
| *email* | user's e-mail |

#### 5.96.3.12 void setFirstName ( String *firstName* )

This sets the user's first name.

**Parameters**

| | |
|---|---|
| *firstName* | user's first name |

#### 5.96.3.13 void setForcePasswordChange ( boolean *forcePasswordChange* ) [protected]

This sets the state indicating if the password is in a state where it needs to be changed.

This is ignored during deserialization, as it should never be set from outside this module. it is never

**Parameters**

| | |
|---|---|
| *forcePassword-Change* | true if it should be set, false if it should be cleared |

#### 5.96.3.14 void setMobile ( String *mobile* )

This sets the user's mobile phone number.

**Precondition**

The mobile parameter should be a String containing only digits.

**Parameters**

| | |
|---|---|
| *mobile* | user's mobile phone number |

#### 5.96.3.15 void setPasswordHash ( String *passwordHash* ) [protected]

This sets the password hash for this user.

It has decreased visibility and is ignored when deserializing requests, as this data should not be set outside this module.

**Parameters**

| | |
|---|---|
| *passwordHash* | hash of the user's password |

#### 5.96.3.16 void setUserType ( UserType *userType* )

This sets the type of the current user.

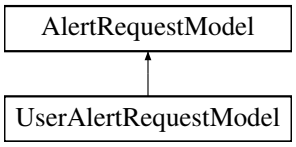
**Parameters**

| | |
|---|---|
| *userType* | user type |

## 5.97 UserAlertRequestModel Class Reference

This model is a JSON representation of the request from User module.

Inheritance diagram for UserAlertRequestModel:




### Additional Inherited Members

### 5.97.1 Detailed Description

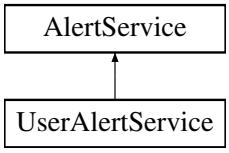
This model is a JSON representation of the request from User module.

During implementation, any additional data that is needed can be added.

## 5.98 UserAlertService Class Reference

Extends AlertService and provides CRUD methods to operate on alerts as well as sends alerts to execute via appropriate strategy.

Inheritance diagram for UserAlertService:




### Public Member Functions

- AlertResponseModel createAlert (AlertRequestModel requestModel)
  *{*
- AlertResponseModel updateAlert (Alert alertModel)
  *{*
- boolean sendAlert ()
  *{*

### Additional Inherited Members

## 5.98.1 Detailed Description

Extends AlertService and provides CRUD methods to operate on alerts as well as sends alerts to execute via appropriate strategy.

This handles all the alerts initiated by User module.

## 5.98.2 Member Function Documentation

#### 5.98.2.1 AlertResponseModel createAlert ( AlertRequestModel *requestModel* ) [virtual]

{

**See Also**

AlertService::createAlert(AlertRequestModel)}

Implements AlertService.

#### 5.98.2.2 boolean sendAlert ( ) [virtual]

{

**See Also**

AlertService::sendAlert()}

Implements AlertService.

#### 5.98.2.3 AlertResponseModel updateAlert ( Alert *alertModel* ) [virtual]

{

**See Also**

AlertService::updateAlert(Alert)}

Implements AlertService.

## 5.99 UserFavoritesList Class Reference

This class ties a userId to that user's ordered list of favorites.

### Public Member Functions

- UserFavoritesList (int userId)

  *This creates a new UserFavoritesList object.*
- int getUserId ()

  *This gets the userId associated with this favorites list.*
- List< FavoriteTransitService > getFavoriteTransitServices ()

  *This retrieves the list of favorite transit services for this user.*
- void setFavoriteTransitServices (List< FavoriteTransitService > favoriteTransitServices)

  *This sets the ordered list of favorite transit services for this user.*

### Private Attributes

- final int **userId**
- List< FavoriteTransitService > **favoriteTransitServices**

### 5.99.1 Detailed Description

This class ties a userId to that user's ordered list of favorites.

### 5.99.2 Constructor & Destructor Documentation

#### 5.99.2.1 UserFavoritesList ( int *userId* )

This creates a new UserFavoritesList object.

Once created, the userId cannot be modified.

**Parameters**

| | |
|---|---|
| *userId* | userId to link this favorites object to |

### 5.99.3 Member Function Documentation

#### 5.99.3.1 List<FavoriteTransitService> getFavoriteTransitServices ( )

This retrieves the list of favorite transit services for this user.

**Returns**

list of favorite transit services

#### 5.99.3.2 int getUserId ( )

This gets the userId associated with this favorites list.

**Returns**

userId

#### 5.99.3.3 void setFavoriteTransitServices ( List< FavoriteTransitService > *favoriteTransitServices* )

This sets the ordered list of favorite transit services for this user.

**Parameters**

| | |
|---|---|
| *favoriteTransitServices* | list to use |

## 5.100 UserFavoritesRepository Class Reference

This class is responsible for handling database access for favorites, and to persist and retrieve UserFavoritesList objects.

### Protected Member Functions

- UserFavoritesList getFavorites (int userId) throws BusBuddyInternalException, BusBuddyNotFoundException

  *This method retrieves the UserFavoritesList object for a given user.*
- void updateFavorites (int userId, UserFavoritesList favorites) throws BusBuddyInternalException, BusBuddyNotFoundException

  *This method updates the UserFavoritesList object for a given user.*

### 5.100.1 Detailed Description

This class is responsible for handling database access for favorites, and to persist and retrieve UserFavoritesList objects.

### 5.100.2 Member Function Documentation

#### 5.100.2.1 UserFavoritesList getFavorites ( int *userId* ) throws BusBuddyInternalException, BusBuddyNotFoundException [protected]

This method retrieves the UserFavoritesList object for a given user.

**Precondition**

The userId passed in must have already saved favorites.

**Parameters**

| | |
|---|---|
| *userId* | User to retrieve favorites for. |

**Returns**

Favorites object for the userId that was passed in.

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is a database error. |
| *BusBuddyNotFoundException* | This exception is thrown if no data has been saved yet, or no such user exists. |

#### 5.100.2.2 void updateFavorites ( int *userId,* UserFavoritesList *favorites* ) throws BusBuddyInternalException, BusBuddyNotFoundException [protected]

This method updates the UserFavoritesList object for a given user.

It creates it if it doesn't exist, and overwrites it if it does.

**Precondition**

The userId must be valid.

**Parameters**

| | |
|---|---|
| *userId* | User to set favorites for. |
| *favorties* | Favorites to set. |

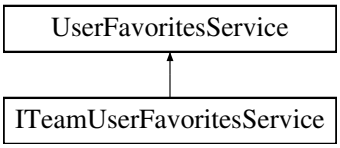
**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is a database error. |
| *BusBuddyNotFound-Exception* | This exception is thrown if no such user exists. |

## 5.101 UserFavoritesService Interface Reference

This is the generic BusBuddy UserFavoritesService interface.

Inheritance diagram for UserFavoritesService:




### Public Member Functions

- UserFavoritesList readFavorites (String sessionToken) throws BusBuddyException

  *This method retrieves the UserFavoritesList object for the current user of a given session.*
- void saveFavorites (String sessionToken, UserFavoritesList favorites) throws BusBuddyException

  *This method updates the UserFavoritesList object for a given user.*

### 5.101.1 Detailed Description

This is the generic BusBuddy UserFavoritesService interface.

This interface contains methods dealing with user favorites. It is one of three interfaces that a user module implementation must implement. It can be implemented as a service or as a service client.

Every method call here will (besides createUser) will result in the session's expiration time being updated due to activity on the session.

### 5.101.2 Member Function Documentation

#### 5.101.2.1 UserFavoritesList readFavorites ( String *sessionToken* ) throws BusBuddyException

This method retrieves the UserFavoritesList object for the current user of a given session.

**Precondition**

The userId linked to the session must have already saved favorites.

**Parameters**

| | |
|---|---|
| *sessionToken* | Session whose user favorites will be retrieved for. |

**Returns**

Favorites object for the userId that was passed in.

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is an internal error. |
| *BusBuddyForbidden-Exception* | This exception is thrown when the current session is invalidated. |
| *BusBuddyNotFound-Exception* | This exception is thrown if no data has been saved yet. |
| *BusBuddyBadRequest-Exception* | This exception is thrown when the sesionToken is blank or empty. |

Implemented in ITeamUserFavoritesService.

#### 5.101.2.2 void saveFavorites ( String *sessionToken,* UserFavoritesList *favorites* ) throws BusBuddyException

This method updates the UserFavoritesList object for a given user.

It creates it if it doesn't exist, and overwrites it if it does.

**Parameters**

| | |
|---|---|
| *sessionToken* | Session whose user the favorites will be retrieved for. |
| *favorties* | Favorites to set. |

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is an internal error. |
| *BusBuddyForbidden-Exception* | This exception is thrown when the current session is invalidated. |
| *BusBuddyBadRequest-Exception* | This exception is thrown when the sesionToken is blank or empty, or the userId on the UserFavoritesList object passed in doesn't match the session. |

Implemented in ITeamUserFavoritesService.

## 5.102 UserInformation Class Reference

UserInformation contains the user related data that we get from User Module.

### Public Member Functions

- int **getUserId** ()
- void **setUserId** (int userId)
- String **getUsername** ()
- void **setUsername** (String username)
- String **getPassword** ()
- void **setPassword** (String password)
- String **getPasswordHash** ()
- void **setPasswordHash** (String passwordHash)
- boolean **isForcePasswordChange** ()
- void **setForcePasswordChange** (boolean forcePasswordChange)
- String **getFirstName** ()
- void **setFirstName** (String firstName)
- String **getEmail** ()
- void **setEmail** (String email)
- Short **getCountryCode** ()
- void **setCountryCode** (Short countryCode)
- String **getMobile** ()
- void **setMobile** (String mobile)
- String **getUserType** ()
- void **setUserType** (String userType)

### Private Attributes

- int **userId**
- String **username**
- String **password**
- String **passwordHash**
- boolean **forcePasswordChange**
- String **firstName**
- String **email**
- Short **countryCode**
- String **mobile**
- String **userType**

#### 5.102.1 Detailed Description

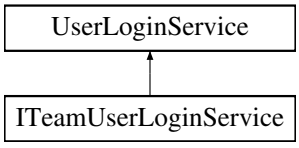
UserInformation contains the user related data that we get from User Module.

This is more or less an exact copy of the User class.

## 5.103 UserLoginService Interface Reference

This is the generic BusBuddy UserLoginService interface.

Inheritance diagram for UserLoginService:



### Public Member Functions

- String login (String username, String password) throws BusBuddyException

  *This method handles the login process.*
- void logout (String sessionToken) throws BusBuddyException

  *This method logs a user out, invalidating their session in the database.*
- User getUser (String sessionToken) throws BusBuddyException

  *This method retrieves the user tied to a supplied session token.*
- String createAlertSession (String sessionToken) throws BusBuddyException

  *This method creates a new session to be used by an alert.*
- void sendUsername (String email) throws BusBuddyException

  *This method sends a user his or her username via e-mail.*
- void sendUsername (short countryCode, String mobile) throws BusBuddyException

  *This method sends a user his or her username via SMS.*
- void resetPassword (String username, String email) throws BusBuddyException

  *This method resets a user's password and sends them their new password via e-mail.*
- void resetPassword (String username, short countryCode, String mobile) throws BusBuddyException

  *This method resets a user's password and sends them their new password via SMS.*

## 5.103.1 Detailed Description

This is the generic BusBuddy UserLoginService interface.

This interface contains methods dealing with user login and session management. It is one of three interfaces that a user module implementation must implement. It can be implemented as a service or as a service client.

## 5.103.2 Member Function Documentation

### 5.103.2.1 String createAlertSession ( String *sessionToken* ) throws BusBuddyException

This method creates a new session to be used by an alert.

Since creation of an alert requires an active user session, this takes an active sessionToken as a parameter. It will then create a new alert session for the same user as the active session. This "alert session" will be long-lived, so it won't expire like the main session. This will allow the Alert module to use this sessionToken when the alert executes.

#### Precondition

The session token must be linked to an active and valid session, which must be linked to an active account.

#### Postcondition

The returned session token points to a valid alert session for this user, which will not expire. The base session's expiration time will be advanced based on this activity against the session.

#### Parameters

| | |
|---|---|
| *sessionToken* | The session token identifying the session that is creating the new alert session. |

#### Returns

Session token representing the new alert seession.

**Exceptions**

| | |
|---|---|
| *BusBuddyBadRequestException* | This exception is thrown if the session token is blank. |
| *BusBuddyForbiddenException* | This exception is thrown if the session token is invalid, linked to an expired session, or the user does not have permission to be signed in. |
| *BusBuddyInternalException* | This exception is thrown if an internal error prevents processing of the request. |

Implemented in ITeamUserLoginService.

#### 5.103.2.2 User getUser ( String *sessionToken* ) throws BusBuddyException

This method retrieves the user tied to a supplied session token.

It will also update the expiration time on the session to keep it valid.

**Precondition**

The session token must be linked to an active and valid session, which must be linked to an active account.

**Postcondition**

The returned session token points to a valid session for this user. The expiration time will be advanced based on this activity against the session.

**Parameters**

| | |
|---|---|
| *sessionToken* | The session token identifying the session that the user information should be retrieved for. |

**Returns**

User object for the user linked to the session represented by the session token parameter.

**Exceptions**

| | |
|---|---|
| *BusBuddyBadRequestException* | This exception is thrown if the session token is blank. |
| *BusBuddyForbiddenException* | This exception is thrown if the session token is invalid, linked to an expired session, or the user does not have permission to be signed in. |
| *BusBuddyInternalException* | This exception is thrown if an internal error prevents processing of the request. |

Implemented in ITeamUserLoginService.

#### 5.103.2.3 String login ( String *username,* String *password* ) throws BusBuddyException

This method handles the login process.

A username and password are supplied. A valid session is created for this user.

**Precondition**

Login credentials must be valid and linked to an active account, or a .common.BusBuddyForbiddenException will be thrown.

**Postcondition**

The returned session token points to a valid session for this user.

**Parameters**

| | |
|---|---|
| *username* | Username of the user to login as. |
| *password* | Password of the user to login as. |

**Returns**

session token of the new session

**Exceptions**

| | |
|---|---|
| *BusBuddyBadRequest-Exception* | This exception is thrown if the username or password are blank. |
| *BusBuddyForbidden-Exception* | This exception is thrown if the credentials are incorrect, or the user does not have permission to sign in. |
| *BusBuddyInternalException* | This exception is thrown if an internal error prevents processing of the request. |

Implemented in ITeamUserLoginService.

#### 5.103.2.4 void logout ( String *sessionToken* ) throws BusBuddyException

This method logs a user out, invalidating their session in the database.

**Precondition**

The sessionToken parameter must be a valid session identifier in the database.

**Postcondition**

The session will be invalidated and future calls using that sessionToken will fail.

**Parameters**

| | |
|---|---|
| *sessionToken* | This is the session token that identifies the session. |

**Exceptions**

| | |
|---|---|
| *BusBuddyNotFound-Exception* | This exception is thrown if the session token is blank or missing on the request.. |
| *BusBuddyNotFound-Exception* | This exception is thrown if the session token is invalid. |
| *BusBuddyInternalException* | This exception is thrown if an internal error prevents processing of the request. |

Implemented in ITeamUserLoginService.

#### 5.103.2.5 void resetPassword ( String *username,* String *email* ) throws BusBuddyException

This method resets a user's password and sends them their new password via e-mail.

**Precondition**

The username and e-mail address provided both link to the same user in the database.

**Postcondition**

The user's password is reset and sent to the user via e-mail.

**Parameters**

| | |
|---|---|
| *username* | Username of the User to reset the password for. |
| *email* | E-mail address of the User to reset the password for. |

**Exceptions**

| | |
|---|---|
| *BusBuddyBadRequest-Exception* | This exception is thrown if the username or e-mail address is blank. |
| *BusBuddyForbidden-Exception* | This exception is thrown if the username and e-mail address combination don't link to a valid user. Also thrown if the account is suspended or deleted. |
| *BusBuddyInternalException* | This exception is thrown if an internal error prevents processing of the request. |

Implemented in ITeamUserLoginService.

#### 5.103.2.6 void resetPassword ( String *username*, short *countryCode*, String *mobile* ) throws BusBuddyException

This method resets a user's password and sends them their new password via SMS.

**Precondition**

The username and mobile information provided both link to the same user in the database.

**Postcondition**

The user's password is reset and sent to the user via SMS.

**Parameters**

| | |
|---|---|
| *username* | Username of the User to reset the password for. |
| *countryCode* | country code of the user's mobile phone number |
| *mobile* | String representing the user's mobile phone number (String should consist entirely of digits) |

**Exceptions**

| | |
|---|---|
| *BusBuddyBadRequest-Exception* | This exception is thrown if the username or mobile number is blank (or non numeric). |
| *BusBuddyForbidden-Exception* | This exception is thrown if the username and mobile information combination don't link to a valid user. Also thrown if the account is suspended or deleted. |
| *BusBuddyInternalException* | This exception is thrown if an internal error prevents processing of the request. |

Implemented in ITeamUserLoginService.

#### 5.103.2.7 void sendUsername ( String *email* ) throws BusBuddyException

This method sends a user his or her username via e-mail.

**Precondition**

The e-mail address provided must be linked to a valid and active account.

**Postcondition**

An e-mail has been sent to the user, containing the user's username.

**Parameters**

| | |
|---|---|
| *email* | E-mail address of the account to send to. |

**Exceptions**

| | |
|---|---|
| *BusBuddyBadRequest-Exception* | This exception is thrown if the e-mail address is blank or invalid. |
| *BusBuddyForbidden-Exception* | This exception is thrown if the e-mail address is linked to an account that is suspended or deleted. |
| *BusBuddyNotFound-Exception* | This exception is thrown if the e-mail address doesn't link to a valid user. |
| *BusBuddyInternalException* | This exception is thrown if an internal error prevents processing of the request. |

Implemented in ITeamUserLoginService.

#### 5.103.2.8 void sendUsername ( short *countryCode,* String *mobile* ) throws BusBuddyException

This method sends a user his or her username via SMS.

**Precondition**

The mobile details provided must be linked to a valid and active account.

**Postcondition**

An SMS has been sent to the user, containing the user's username.

**Parameters**

| | |
|---|---|
| *email* | E-mail address of the account to send to. |

**Exceptions**

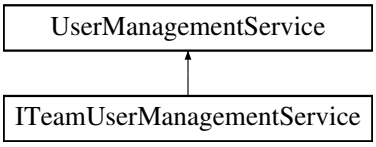
| | |
|---|---|
| *BusBuddyBadRequest-Exception* | This exception is thrown if the mobile is blank or non-numeric. |
| *BusBuddyForbidden-Exception* | This exception is thrown if the mobile information is linked to an account that is suspended or deleted. |
| *BusBuddyNotFound-Exception* | This exception is thrown if the mobile information doesn't link to a valid user. |
| *BusBuddyInternalException* | This exception is thrown if an internal error prevents processing of the request. |

Implemented in ITeamUserLoginService.

## 5.104 UserManagementService Interface Reference

This is the generic BusBuddy UserManagementService interface.

Inheritance diagram for UserManagementService:




## Public Member Functions

- User createUser (User userToCreate, String password) throws BusBuddyException

  *This method creates a user in the database.*
- User findUserByUsername (String sessionToken, String username) throws BusBuddyException

  *This method attempts to retrieve a user by username.*
- User findUserByEmail (String sessionToken, String email) throws BusBuddyException

  *This method attempts to retrieve a user by e-mail address.*
- User findUserByMobile (String sessionToken, short countryCode, String mobile) throws BusBuddyException

  *This method attempts to retrieve a user by mobile phone number.*
- void updateUser (String sessionToken, User newUserData, String password) throws BusBuddyException

  *This method updates a user in the database.*
- void deleteUser (String sessionToken, User userToDelete) throws BusBuddyException

  *This method deletes a user from the database.*

### 5.104.1 Detailed Description

This is the generic BusBuddy UserManagementService interface.

This interface contains methods dealing with user account management. It is one of three interfaces that a user module implementation must implement. It can be implemented as a service or as a service client.

Every method call here will (besides createUser) will result in the session's expiration time being updated due to activity on the session.

### 5.104.2 Member Function Documentation

#### 5.104.2.1 User createUser ( User *userToCreate,* String *password* ) throws BusBuddyException

This method creates a user in the database.

**Precondition**

No other user with this username, e-mail address, or mobile phone exists.

**Postcondition**

User is created with the given user data.

**Parameters**

| | |
|---|---|
| *user* | User data of the user to create (the ID will be ignored). |
| *password* | Password of the user to create. |

**Returns**

new user object

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when an internal error prevents creation of the user. |
| *BusBuddyConflictException* | This exception is thrown when the requested user record would create a duplicate username, e-mail address, or mobile phone data in the database. |

Implemented in ITeamUserManagementService.

#### 5.104.2.2 void deleteUser ( String *sessionToken,* User *userToDelete* ) throws BusBuddyException

This method deletes a user from the database.

It will delete the user with the same ID as the user passed in as a parameter.

**Precondition**

A user with the specified user ID on the User object must already exist.

**Postcondition**

User object in database will be deleted.

**Parameters**

| | |
|---|---|
| *sessionToken* | session token for the currently logged in user |
| *userToDelete* | This user object should have the same ID as the user to be deleted. |

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is an internal error preventing execution of the request. |
| *BusBuddyForbiddenException* | This exception is thrown if the session token is invalid, linked to an expired session, or the user does not have permission to make this change. |
| *BusBuddyNotFoundException* | This exception is thrown if the targeted user is not found in the database. |

Implemented in ITeamUserManagementService.

#### 5.104.2.3 User findUserByEmail ( String *sessionToken,* String *email* ) throws BusBuddyException

This method attempts to retrieve a user by e-mail address.

It is not case sensitive. The method will take an e-mail address, read the details from the database, and construct a user object with the given details.

**Precondition**

A user with the supplied e-mail address exists within the database.

**Postcondition**

A user will be returned whose e-mail address matches the supplied e-mail address parameter.

**Parameters**

| | |
|---|---|
| *email* | This is the e-mail address to look up. |

**Returns**

The user with the given e-mail address.

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | An internal error prevents execution of the request. |
| *BusBuddyForbidden-Exception* | The currently logged in user does not have permission to view the result of this search. |
| *BusBuddyNotFound-Exception* | This exception is thrown when the requested user record could not be found. |

Implemented in ITeamUserManagementService.

#### 5.104.2.4 User findUserByMobile ( String *sessionToken,* short *countryCode,* String *mobile* ) throws BusBuddyException

This method attempts to retrieve a user by mobile phone number.

The method will take a mobile phone number, read the details from the database, and construct a user object with the given details.

**Precondition**

A user with the supplied mobile phone number exists within the database.

**Postcondition**

A user will be returned whose mobile phone details match the supplied parameters.

**Parameters**

| | |
|---|---|
| *countryCode* | This is the country code of the user's mobile phone number. |
| *mobile* | This is the remainder of the user's mobile phone number. This string should consist entirely of digits. |

**Returns**

The user with the given mobile phone details.

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | An internal error prevents execution of the request. |
| *BusBuddyForbidden-Exception* | The currently logged in user does not have permission to view the result of this search. |
| *BusBuddyNotFound-Exception* | This exception is thrown when the requested user record could not be found. |

Implemented in ITeamUserManagementService.

#### 5.104.2.5 User findUserByUsername ( String *sessionToken,* String *username* ) throws BusBuddyException

This method attempts to retrieve a user by username.

It is not case sensitive. The method will take a username, read the details from the database, and construct a user object with the given details.

**Precondition**

A user with the supplied username exists within the database.

**Postcondition**

A user will be returned whose username matches the supplied username parameter.

**Parameters**

| | |
|---|---|
| *username* | This is the username to look up. |

**Returns**

The user with the given username.

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | An internal error prevents execution of the request. |
| *BusBuddyForbidden-Exception* | The currently logged in user does not have permission to view the result of this search. |
| *BusBuddyNotFound-Exception* | This exception is thrown when the requested user record could not be found. |

Implemented in ITeamUserManagementService.

#### 5.104.2.6 void updateUser ( String *sessionToken,* User *newUserData,* String *password* ) throws BusBuddyException

This method updates a user in the database.

It will update the user with the same ID as the user passed in as a parameter. The username will not be updated, but all other fields will be.

**Precondition**

A user with the specified user ID on the User object must already exist.

**Postcondition**

User object in database will be updated with the data from the parameter User object.

**Parameters**

| | |
|---|---|
| *sessionToken* | session token for the currently logged in user |
| *newUserData* | User object with the new user data on it |
| *password* | New password to set on the user. |

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is an internal error preventing execution of the request. |
| *BusBuddyForbiddenException* | This exception is thrown if the session token is invalid, linked to an expired session, or the user does not have permission to make this change. |
| *BusBuddyNotFoundException* | This exception is thrown when the targeted user to receive the update does not exist. |
| *BusBuddyConflictException* | This exception is thrown when the requested change would create a duplicate mobile phone or e-mail address in the database. |

Implemented in ITeamUserManagementService.

## 5.105 UserRepository Class Reference

This class is responsible for handling database access for User objects, and to construct, persist, and retrieve User objects.

### Package Functions

- User createUser (User user, String password) throws BusBuddyConflictException, BusBuddyInternalException

  *This method creates a user in the database.*
- User getUserById (int userId) throws BusBuddyInternalException, BusBuddyNotFoundException

  *This method attempts to retrieve a user by id number.*
- User getUserByUsername (String username) throws BusBuddyInternalException, BusBuddyNotFoundException

  *This method attempts to retrieve a user by username.*
- User getUserByEmail (String email) throws BusBuddyInternalException, BusBuddyNotFoundException

  *This method attempts to retrieve a user by e-mail address.*
- User getUserByMobile (short countryCode, String mobile) throws BusBuddyInternalException, BusBuddyNotFoundException

  *This method attempts to retrieve a user by mobile phone number.*
- void updateUser (User newUserData) throws BusBuddyInternalException, BusBuddyNotFoundException, - BusBuddyConflictException

  *This method updates a user in the database.*
- void deleteUser (User userToDelete) throws BusBuddyInternalException, BusBuddyNotFoundException

  *This method deletes a user from the database.*

### 5.105.1 Detailed Description

This class is responsible for handling database access for User objects, and to construct, persist, and retrieve User objects.

### 5.105.2 Member Function Documentation

#### 5.105.2.1 User createUser ( User *user,* String *password* ) throws BusBuddyConflictException, BusBuddyInternalException [package]

This method creates a user in the database.

**Precondition**

No other user with this username, e-mail address, or mobile phone exists.

**Postcondition**

User is created with the given user data.

**Parameters**

| | |
|---|---|
| *user* | User data of the user to create (the ID will be ignored). |
| *password* | Password of the user to create. |

**Returns**

new user object

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is a database error. |
| *BusBuddyConflictException* | This exception is thrown when the requested user record would create a duplicate username, e-mail address, or mobile phone data in the database. |

#### 5.105.2.2 void deleteUser ( User *userToDelete* ) throws BusBuddyInternalException, BusBuddyNotFoundException [package]

This method deletes a user from the database.

It will delete the user with the same ID as the user passed in as a parameter.

**Precondition**

A user with the specified user ID on the User object must already exist.

**Postcondition**

User object in database will be deleted.

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is a database error. |
| *BusBuddyNotFound-Exception* | This exception is thrown when the requested user record could not be found. |

#### 5.105.2.3 User getUserByEmail ( String *email* ) throws BusBuddyInternalException, BusBuddyNotFoundException [package]

This method attempts to retrieve a user by e-mail address.

It is not case sensitive. The method will take an e-mail address, read the details from the database, and construct a user object with the given details.

**Precondition**

A user with the supplied e-mail address exists within the database.

**Postcondition**

A user will be returned whose e-mail address matches the supplied e-mail address parameter.

**Parameters**

| | |
|---|---|
| *email* | This is the e-mail address to look up. |

**Returns**

The user with the given e-mail address.

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is a database error. |
| *BusBuddyNotFound-Exception* | This exception is thrown when the requested user record could not be found. |

#### 5.105.2.4 User getUserById ( int *userId* ) throws BusBuddyInternalException, BusBuddyNotFoundException [package]

This method attempts to retrieve a user by id number.

The method will take a user id, read the details from the database, and construct a user object with the given details.

**Precondition**

A user with the supplied user id exists within the database.

**Postcondition**

A user will be returned whose user id matches the supplied userId parameter.

**Parameters**

| | |
|---|---|
| *userId* | This is the user ID to look up. |

**Returns**

The user with the given ID.

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is a database error. |
| *BusBuddyNotFound-Exception* | This exception is thrown when the requested user record could not be found. |

#### 5.105.2.5 User getUserByMobile ( short *countryCode,* String *mobile* ) throws BusBuddyInternalException, BusBuddyNotFoundException [package]

This method attempts to retrieve a user by mobile phone number.

The method will take a mobile phone number, read the details from the database, and construct a user object with the given details.

**Precondition**

A user with the supplied mobile phone number exists within the database.

**Postcondition**

A user will be returned whose mobile phone details match the supplied parameters.

**Parameters**

| | |
|---|---|
| *countryCode* | This is the country code of the user's mobile phone number. |
| *mobile* | This is the remainder of the user's mobile phone number. This string should consist entirely of digits. |

**Returns**

The user with the given mobile phone details.

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is a database error. |
| *BusBuddyNotFound-Exception* | This exception is thrown when the requested user record could not be found. |

#### 5.105.2.6 User getUserByUsername ( String *username* ) throws BusBuddyInternalException, BusBuddyNotFoundException [package]

This method attempts to retrieve a user by username.

It is not case sensitive. The method will take a username, read the details from the database, and construct a user object with the given details.

**Precondition**

A user with the supplied username exists within the database.

**Postcondition**

A user will be returned whose username matches the supplied username parameter.

**Parameters**

| | |
|---|---|
| *username* | This is the username to look up. |

**Returns**

The user with the given username.

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is a database error. |
| *BusBuddyNotFound-Exception* | This exception is thrown when the requested user record could not be found. |

#### 5.105.2.7 void updateUser ( User *newUserData* ) throws BusBuddyInternalException, BusBuddyNotFoundException, BusBuddyConflictException [package]

This method updates a user in the database.

It will update the user with the same ID as the user passed in as a parameter. The username will not be updated, but all other fields will be.

**Precondition**

A user with the specified user ID on the User object must already exist.

**Postcondition**

User object in database will be updated with the data from the parameter User object.

**Parameters**

| | |
|---|---|
| *newUserData* | User object with the new user data on it |

**Exceptions**

| | |
|---|---|
| *BusBuddyInternalException* | This exception is thrown when there is a database error. |
| *BusBuddyNotFound-Exception* | This exception is thrown when the requested user record could not be found. |
| *BusBuddyConflictException* | This exception is thrown when the requested user record would create a duplicate mobile phone or e-mail address in the database. |

## 5.106 UserSessionInformation Class Reference

A model that stores all the information needed to call user module about user information.

### Public Member Functions

- String **getUserId** ()
- void **setUserId** (String userId)
- String **getUserSessionToken** ()
- void **setUserSessionToken** (String userSessionToken)

### Private Attributes

- String userId

  *User ID of the user that initiated the alert.*
- String userSessionToken

  *A long lived session token to validate the authenticity of request to UserModule.*

### 5.106.1 Detailed Description

A model that stores all the information needed to call user module about user information.

### 5.106.2 Member Data Documentation

#### 5.106.2.1 String userSessionToken [private]

A long lived session token to validate the authenticity of request to UserModule.

This is required to get the current user information in order to alert the user.

## 5.107 UserTrackingAlertObject Class Reference

User tracking alert information obtained from the user interface when the user registers for an alert.

### Public Member Functions

- URL **getTransitCoInfo** ()
- int **getRouteID** ()

### Private Attributes

- URL transitCoInfo
  *URL uniquely identifies transit company information.*
- int routeID
  *Route number user is watching for vehicles.*
- Location stopLocation
  *GPS coordinates of vehicle stop closest to user.*
- Date scheduledTime
  *Time vehicle is scheduled to be at closest stop.*
- Date alertTime
  *Amount of lead time user needs to get to the vehicle stop.*
- String userContactInfo
  *How to contact a user with an alert.*
- AlertType type
  *Type of alert the user is registered.*

### 5.107.1 Detailed Description

User tracking alert information obtained from the user interface when the user registers for an alert.

Primarily, this meta data will come from the user interface and stored in user module.

### 5.107.2 Member Data Documentation

#### 5.107.2.1 Date alertTime [private]

Amount of lead time user needs to get to the vehicle stop.

Obtained from the user interface when the user signs up for an alert.

#### 5.107.2.2 int routeID [private]

Route number user is watching for vehicles.

Obtained by translating user interface route description drop down to transit information route id.

#### 5.107.2.3 Date scheduledTime [private]

Time vehicle is scheduled to be at closest stop.

Obtained from information uploaded by a transit company. Used for determining delay alerts.

#### 5.107.2.4 Location stopLocation [private]

GPS coordinates of vehicle stop closest to user.

Obtained from information uploaded to transit module by a transit company.

#### 5.107.2.5 URL transitCoInfo [private]

URL uniquely identifies transit company information.

The transit company URL information to distinguish this vehicle from other vehicles in different cities with similar route numbers. Obtained from the user interface when the user registers for an alert, selected from a drop down derived from metadata {}.

#### 5.107.2.6 AlertType type [private]

Type of alert the user is registered.

A user may sign up for tracking alerts when a vehicle is approaching their stop. Transit companies may sign up for delay alerts when their vehicle(s) are running behind schedule.

#### 5.107.2.7 String userContactInfo [private]

How to contact a user with an alert.

User contact information (phone number or URL) where to send tracking alerts. Obtained from user interface when user signs up for an alert. On Bus Buddy system start, obtained from alert repository.

## 5.108 UserType Enum Reference

This is an enumeration of the different statuses that a user can be assigned.

### Public Attributes

- NORMAL_USER

  *This is a standard user.*
- SYSTEM_ADMINISTRATOR

  *This is a system administrator, who can read and write to other users' data.*
- SUSPENDED_USER

  *A user account marked as suspended cannot create or use sessions.*

### 5.108.1 Detailed Description

This is an enumeration of the different statuses that a user can be assigned.

## 5.109 VehicleObject Class Reference

Value Object containing vehicle information obtained when the user registers a vehicle using the user interface.

### Public Member Functions

- String **getGPSDeviceInfo** ()
- URL **getTransitCoURL** ()
- int **getGPSDeviceID** ()
- int **getCurrentRoute** ()

### Private Attributes

- int gpsDeviceID

  *GPS hardware device ID.*
- String gpsDeviceInfo

  *GPS device contact information, commercial web URL, GPS wireless connection or port number.*
- URL transitCoURL

  *Transit company operating this vehicle.*
- int currentRoute

  *Current route number.*

### 5.109.1 Detailed Description

Value Object containing vehicle information obtained when the user registers a vehicle using the user interface.

### 5.109.2 Member Data Documentation

#### 5.109.2.1 int currentRoute [private]

Current route number.

Obtain and updated by the user interface.

#### 5.109.2.2 int gpsDeviceID [private]

GPS hardware device ID.

Hardware GPS device ID, uniquely identifies a GPS unit. Obtained from user interface when a vehicle registers for tracking.

#### 5.109.2.3 String gpsDeviceInfo [private]

GPS device contact information, commercial web URL, GPS wireless connection or port number.

Obtained from user interface when a vehicle is registered for tracking.

#### 5.109.2.4 URL transitCoURL [private]

Transit company operating this vehicle.

Obtained from the information uploaded to TransitInfo by the transit company. User selects transit company name from a drop down on the user interface when registering a vehicle.

## 5.110 VehicleRepository Class Reference

Repository for information on vehicles registered on a route.

### Public Member Functions

- VehicleRepository ()

  *Create the initial repository for saving vehicles registered with the tracking service.*
- void addVehicle (TransitVehicle vehicle)

  *Add a vehicle to the repository.*
- void removeVehicle (int gpsDeviceID)

  *Remove a vehicle from the repository.*
- void updateVehicle (TransitVehicle vehicle)

  *A vehicle may switch routes, update an existing vehicle in the repository.*
- TransitVehicle findVehicle (int gpsDeviceID)

  *Find a vehicle currently stored in the repository based on the unique GPS device ID.*

### Static Public Member Functions

- static ArrayList< TransitVehicle > findVehiclesByRoute (URL transitCoURL, int routeID)

  *Find all vehicles from a transit company registered on a route.*

### Private Attributes

- ArrayList< TransitVehicle > vehicleList = null

  *List of vehicles currently registered and available in this repository.*

### 5.110.1 Detailed Description

Repository for information on vehicles registered on a route.

### 5.110.2 Member Function Documentation

#### 5.110.2.1 TransitVehicle findVehicle ( int *gpsDeviceID* )

Find a vehicle currently stored in the repository based on the unique GPS device ID.

**Parameters**

| | |
|---|---|
| *gpsDeviceID* | - integer GPS device ID |

**Returns**

VehicleObject matching vehicle or null if no matching vehicle found.

#### 5.110.2.2 static ArrayList<TransitVehicle> findVehiclesByRoute ( URL *transitCoURL*, int *routeID* ) [static]

Find all vehicles from a transit company registered on a route.

**Parameters**

| | |
|---|---|
| *transitCoURL* | URL or the transit company |
| *routeID* | - integer route that vehicle is currently registered on. |

**Returns**

ArrayList<TransitVehicle> of all vehicles for transit company registered on the route or null if no matching vehicles found.

#### 5.110.2.3 void removeVehicle ( int *gpsDeviceID* )

Remove a vehicle from the repository.

**Parameters**

| | |
|---|---|
| *gpsDeviceID* | - integer the GPS id if the vehicle to remove. |

#### 5.110.2.4 void updateVehicle ( TransitVehicle *vehicle* )

A vehicle may switch routes, update an existing vehicle in the repository.

**Parameters**

| | |
|---|---|
| *vehicle* | - VehicleObject new vehicle information from driver through UI |

# Index


AbstractFeedParserTemplate, 23
addAlertSpecification
    tracking::TransitVehicle, 123
addUserTrackingAlert
    tracking::ITrackingService, 81
    tracking::TrackingServiceController, 113
Alert, 25
alert.client, 13
alert.client.model, 13
alert.controller, 14
alert.controller.model, 14
alert.domain, 15
alert.domain.model, 15
alert.enums, 16
alert.service, 16
alert::client::AlertUserClient
    getuserInformation, 44
    verifySessionToken, 44
alert::controller::AlertRequestController
    processTrackingAlertRequest, 35
    processTransitAlertRequest, 35
    processUserAlertRequest, 35
    verifySession, 36
alert::controller::CertificateHandler
    verifySessionToken, 52
alert::controller::ISessionHandler
    verifySessionToken, 71
alert::controller::SessionTokenHandler
    userClient, 102
    verifySessionToken, 102
alert::controller::SessionVerificationFactory
    getSessionTokenVerificationStrategy, 103
alert::controller::model::AlertRequestModel
    alertInitiator, 37
alert::domain::AlertFactory
    createAlert, 29
alert::domain::AlertRepository
    deleteAlert, 32
    getAlertByDateTime, 32
    getAlertByRoute, 33
    getAlertByUserId, 33
    saveAlert, 33
    updateAlert, 33
alert::domain::model::Alert
    alertInitiator, 27
    alertRunType, 27
    alertType, 27
    Status, 27
    userInformation, 27
alert::domain::model::OneTimeAlert
    dateExecuted, 84
alert::domain::model::RecurringAlert
    alertRecurringType, 87
    repeatEvery, 87
alert::domain::model::RecurringData
    dayOfMonth, 88
    dayOfWeek, 88
    dayOfYear, 88
    startHour, 88
    startMinute, 89
alert::domain::model::TripInformation
    getRouteIds, 125
    tripData, 126
alert::domain::model::UserSessionInformation
    userSessionToken, 152
alert::enums::AlertNotificationType
    PlannedDisruption, 30
    ScheduleInformation, 30
    UnplannedDisruption, 30
alert::enums::AlertStatus
    Deactive, 43
    Error, 43
    Expired, 43
alert::service::AlertExecuteStrategyFactory
    getAlertExecuteStrategy, 28
alert::service::AlertExecutionScheduler
    getAlertServices, 28
    runAlert, 28
alert::service::AlertService
    alertExecuteStrategyFactory, 40
    alertFactory, 40
    alertRepository, 40
    createAlert, 38
    deleteAlert, 39
    saveAlert, 39
    sendAlert, 39
    updateAlert, 39
alert::service::AlertServiceFactory
    getAlertService, 41
    trackingAlertService, 41
    transitAlertService, 41
    userAlertService, 41
alert::service::IAlertExecuteStrategy
    execute, 69
alert::service::RouteAlertExecuteStrategy
    alertRepository, 91
    execute, 91
    userClient, 91

alert::service::ScheduleAlertExecuteStrategy
alertRepository, 97
userClient, 97
alert::service::TrackingAlertService
createAlert, 109
sendAlert, 109
updateAlert, 109
alert::service::TransitAlertService
createAlert, 115
sendAlert, 115
updateAlert, 115
alert::service::UserAlertService
createAlert, 132
sendAlert, 132
updateAlert, 132
AlertExecuteStrategyFactory, 27
alertExecuteStrategyFactory
alert::service::AlertService, 40
AlertExecutionScheduler, 28
AlertFactory, 29
alertFactory
alert::service::AlertService, 40
AlertInitiator, 30
alertInitiator
alert::controller::model::AlertRequestModel, 37
alert::domain::model::Alert, 27
alertList
tracking::BusVehicle, 51
AlertNotificationType, 30
AlertRangeLogic, 31
AlertRecurringType, 31
alertRecurringType
alert::domain::model::RecurringAlert, 87
AlertRepository, 32
alertRepository
alert::service::AlertService, 40
alert::service::RouteAlertExecuteStrategy, 91
alert::service::ScheduleAlertExecuteStrategy, 97
AlertRequestController, 34
alertRequestController
transit::ITeamTransitServiceController, 74
AlertRequestModel, 36
AlertResponseModel, 37
AlertRunType, 37
alertRunType
alert::domain::model::Alert, 27
AlertService, 37
AlertServiceFactory, 40
AlertSpecification, 41
AlertStatus, 42
alertTime
tracking::UserTrackingAlertObject, 152
AlertType, 43
alertType
alert::domain::model::Alert, 27
AlertUserClient, 43
and
common::Specification< T >, 103
BaseController, 44
BusBuddyBadRequestException, 46
BusBuddyConflictException, 46
BusBuddyException, 47
BusBuddyForbiddenException, 48
BusBuddyInternalException, 49
BusBuddyNotFoundException, 50
BusVehicle, 51
calculateTrip
transit::ITeamTripServiceController, 75
transit::TripService, 126
cause
transit::Detour, 55
CertificateHandler, 52
checkForAlerts
tracking::TransitVehicle, 123
checkPermissions
user::ITeamUserLoginService, 77
CommercialTracking, 52
tracking::CommercialTracking, 53
CommercialTracking.CommercialTrackingHolder, 54
common, 17
common::BaseController
handleBusBuddyException, 45
handleGenericException, 45
common::BusBuddyBadRequestException
getHttpCode, 46
common::BusBuddyConflictException
getHttpCode, 47
common::BusBuddyException
getHttpCode, 48
common::BusBuddyForbiddenException
getHttpCode, 49
common::BusBuddyInternalException
getHttpCode, 50
common::BusBuddyNotFoundException
getHttpCode, 51
common::HashUtility
hash, 69
common::MessageDeliveryUtility
sendEmail, 83
sendSms, 83
common::Specification< T >
and, 103
isSatisfiedBy, 104
not, 104
or, 104
convertJSONAlertInput
tracking::TrackingResponseModel, 112
convertJSONVehicleInput
tracking::TrackingResponseModel, 112
covered
transit::Stop, 106
createAlert
alert::domain::AlertFactory, 29
alert::service::AlertService, 38
alert::service::TrackingAlertService, 109
alert::service::TransitAlertService, 115

alert::service::UserAlertService, 132
createAlertObserver
tracking::TrackingAlertFactory, 107
createAlertSession
user::ITeamUserLoginService, 78
user::UserLoginService, 138
createSession
user::SessionRepository, 100
createTransitVehicle
tracking::TransitVehicleFactory, 123
createUser
user::ITeamUserManagementService, 80
user::UserManagementService, 143
user::UserRepository, 147
currentRoute
tracking::VehicleObject, 154

dateExecuted
alert::domain::model::OneTimeAlert, 84
dayOfMonth
alert::domain::model::RecurringData, 88
dayOfWeek
alert::domain::model::RecurringData, 88
dayOfYear
alert::domain::model::RecurringData, 88
Deactive
alert::enums::AlertStatus, 43
DelayAlertLogic, 54
delete
transit::RouteRepository, 94
deleteAlert
alert::domain::AlertRepository, 32
alert::service::AlertService, 39
deleteUser
user::ITeamUserManagementService, 80
user::UserManagementService, 144
user::UserRepository, 148
description
transit::Stop, 106
Detour, 54
transit::Detour, 55
detourId
transit::Detour, 55
detours
transit::Route, 90
discountedFare
transit::Fare, 57

Error
alert::enums::AlertStatus, 43
estimatedDelay
transit::Detour, 56
execute
alert::service::IAlertExecuteStrategy, 69
alert::service::RouteAlertExecuteStrategy, 91
Expired
alert::enums::AlertStatus, 43

Fare, 56
transit::Fare, 56, 57
FavoriteTransitService, 57
user::FavoriteTransitService, 58
findUserByEmail
user::ITeamUserManagementService, 80
user::UserManagementService, 144
findUserByMobile
user::ITeamUserManagementService, 80
user::UserManagementService, 145
findUserByUsername
user::ITeamUserManagementService, 80
user::UserManagementService, 145
findVehicle
tracking::VehicleRepository, 155
findVehiclesByRoute
tracking::VehicleRepository, 155
fireRouteDistruptionEvent
transit::TransitProvider, 118
formatJSONResponse
tracking::TrackingResponseModel, 112

GPSLocationObject, 61
GPSLocationObserver, 61
GPSLocationTracking, 63
GPSPuller, 64
tracking::GPSPuller, 65
GPSPuller.GPSPullerHolder, 65
GPSPusher, 65
tracking::GPSPusher, 66
GPSPusher.GPSPusherHolder, 67
GPSVehicleTracker, 67
tracking::GPSVehicleTracker, 68
GTFSFeedParser, 68
getAlertByDateTime
alert::domain::AlertRepository, 32
getAlertByRoute
alert::domain::AlertRepository, 33
getAlertByUserId
alert::domain::AlertRepository, 33
getAlertExecuteStrategy
alert::service::AlertExecuteStrategyFactory, 28
getAlertService
alert::service::AlertServiceFactory, 41
getAlertServices
alert::service::AlertExecutionScheduler, 28
getAll
transit::RouteRepository, 94
getCountryCode
user::User, 128
getCreationTime
user::Session, 98
getEmail
user::User, 128
getExpirationTime
user::Session, 98
getFavoriteRouteIds
user::FavoriteTransitService, 58
getFavoriteTransitServices
user::UserFavoritesList, 133

getFavorites
user::UserFavoritesRepository, 134
getFirstName
user::User, 128
getGPSLocation
tracking::GPSLocationObserver, 62
getGPSTypeFromURL
tracking::TransitVehicleFactory, 123
getHttpCode
common::BusBuddyBadRequestException, 46
common::BusBuddyConflictException, 47
common::BusBuddyException, 48
common::BusBuddyForbiddenException, 49
common::BusBuddyInternalException, 50
common::BusBuddyNotFoundException, 51
getInstance
tracking::CommercialTracking, 53
tracking::GPSPusher, 66
getMobile
user::User, 129
getPasswordHash
user::User, 129
getRoute
transit::GoogleTransitServiceAdapter, 60
transit::ITeamTransitServiceController, 73
transit::PersistedTransitFeed, 85
transit::TransitFeed, 116
transit::TransitService, 120
getRouteIds
alert::domain::model::TripInformation, 125
getRoutes
transit::GoogleTransitServiceAdapter, 60
transit::ITeamTransitServiceController, 73
transit::PersistedTransitFeed, 86
transit::TransitFeed, 116
transit::TransitService, 121
getServiceURL
transit::ITeamTransitServiceController, 74
transit::TransitService, 121
getSession
user::SessionRepository, 100
getSessionToken
user::Session, 98
getSessionTokenVerificationStrategy
alert::controller::SessionVerificationFactory, 103
getSpec
tracking::TrackingAlertObserver, 108
getStopTimes
transit::Stop, 105
getTransitServiceUrl
user::FavoriteTransitService, 58
getTransitVehicleLocation
tracking::ITrackingService, 81
getTransitInfo
transit::ITeamTransitServiceController, 74
transit::TransitService, 121
getUser
user::ITeamUserLoginService, 78
user::UserLoginService, 139
getUserByEmail
user::UserRepository, 148
getUserById
user::UserRepository, 149
getUserByMobile
user::UserRepository, 149
getUserByUsername
user::UserRepository, 150
getUserId
user::Session, 99
user::User, 129
user::UserFavoritesList, 133
getUserType
user::User, 129
getUsername
user::User, 129
getVehicleGPSDeviceID
tracking::TransitVehicleFactory, 124
getuserInformation
alert::client::AlertUserClient, 44
GoogleTransitServiceAPI, 61
GoogleTransitServiceAdapter, 59
transit::GoogleTransitServiceAdapter, 60
gpsDeviceID
tracking::VehicleObject, 154
gpsDeviceInfo
tracking::VehicleObject, 154
gpsUpdate
tracking::GPSLocationObserver, 62

handleBusBuddyException
common::BaseController, 45
handleGenericException
common::BaseController, 45
handleRouteDisruptionEvent
transit::ITeamTransitServiceController, 74
transit::TransitProviderObserver, 120
hash
common::HashUtility, 69
HashUtility, 68

IAlertExecuteStrategy, 69
ISessionHandler, 71
ITeamTransitServiceController, 72
ITeamTripServiceController, 75
ITeamUserFavoritesService, 76
ITeamUserLoginService, 77
ITeamUserManagementService, 79
ITrackingService, 80
inAlertRange
tracking::AlertSpecification, 42
InvalidRouteParseException, 70
transit::InvalidRouteParseException, 71
isAlertSession
user::Session, 99
isFavoriteTransitService
user::FavoriteTransitService, 58
isForcePasswordChange

user::User, 129
isSatisfiedBy
common::Specification< T >, 104
transit::RouteSpecification, 96
isValid
user::Session, 99

killSession
user::SessionRepository, 101

loadFeed
transit::AbstractFeedParserTemplate, 24
Location, 82
transit::Location, 83
login
user::ITeamUserLoginService, 78
user::UserLoginService, 139
logo
transit::TransitInfo, 117
logout
user::ITeamUserLoginService, 78
user::UserLoginService, 140

MessageDeliveryUtility, 83

name
transit::TransitProvider, 119
not
common::Specification< T >, 104

OneTimeAlert, 84
or
common::Specification< T >, 104

parseFeed
transit::AbstractFeedParserTemplate, 24
transit::GTFSFeedParser, 68
PersistedTransitFeed, 85
PlannedDisruption
alert::enums::AlertNotificationType, 30
processTrackingAlertRequest
alert::controller::AlertRequestController, 35
processTransitAlertRequest
alert::controller::AlertRequestController, 35
processUserAlertRequest
alert::controller::AlertRequestController, 35
providerId
transit::TransitProvider, 119

read
transit::RouteRepository, 94
readFavorites
user::ITeamUserFavoritesService, 76
user::UserFavoritesService, 135
RecurringAlert, 86
RecurringData, 87
registerGPSDevice
tracking::GPSLocationTracking, 63
registerObserver
transit::TransitProvider, 119
registerVehicleOnRoute
tracking::ITrackingService, 81
tracking::TrackingServiceController, 113
removeVehicle
tracking::VehicleRepository, 156
repeatEvery
alert::domain::model::RecurringAlert, 87
resetPassword
user::ITeamUserLoginService, 78
user::UserLoginService, 140, 141
Route, 89
transit::Route, 90
RouteAlertExecuteStrategy, 90
routeBatch
transit::InvalidRouteParseException, 71
RouteDisruptionAlert, 92
RouteDisruptionEvent, 92
transit::RouteDisruptionEvent, 93
routeID
tracking::UserTrackingAlertObject, 152
routeId
transit::RouteDisruptionAlert, 92
routeName
transit::Route, 90
RouteRepository, 93
RouteSpecification, 95
runAlert
alert::service::AlertExecutionScheduler, 28

save
transit::RouteRepository, 94, 95
saveAlert
alert::domain::AlertRepository, 33
alert::service::AlertService, 39
saveFavorites
user::ITeamUserFavoritesService, 76
user::UserFavoritesService, 136
saveRoutes
transit::AbstractFeedParserTemplate, 24
ScheduleAlertExecuteStrategy, 96
ScheduleInformation
alert::enums::AlertNotificationType, 30
scheduledTime
tracking::UserTrackingAlertObject, 152
sendAlert
alert::service::AlertService, 39
alert::service::TrackingAlertService, 109
alert::service::TransitAlertService, 115
alert::service::UserAlertService, 132
sendEmail
common::MessageDeliveryUtility, 83
sendSms
common::MessageDeliveryUtility, 83
sendUsername
user::ITeamUserLoginService, 79
user::UserLoginService, 141, 142
serialVersionUID
transit::InvalidRouteParseException, 71
Session, 97

user::Session, 98
SessionRepository, 99
SessionTokenHandler, 101
SessionVerificationFactory, 102
setCountryCode
user::User, 130
setDiscountedFare
transit::Fare, 57
setEmail
user::User, 130
setExpirationTime
user::Session, 99
setFavoriteRouteIds
user::FavoriteTransitService, 59
setFavoriteTransitService
user::FavoriteTransitService, 59
setFavoriteTransitServices
user::UserFavoritesList, 133
setFirstName
user::User, 130
setForcePasswordChange
user::User, 130
setGPSLocation
tracking::GPSLocationObserver, 63
setMobile
user::User, 130
setPasswordHash
user::User, 131
setRegularFare
transit::Fare, 57
setSpec
tracking::TrackingAlertObserver, 108
setStops
transit::Route, 90
setUserType
user::User, 131
setValid
user::Session, 99
Specification< T >, 103
start
transit::AbstractFeedParserTemplate, 25
startHour
alert::domain::model::RecurringData, 88
startMinute
alert::domain::model::RecurringData, 89
startTrackingController
tracking::ITrackingService, 82
Status
alert::domain::model::Alert, 27
Stop, 104
transit::Stop, 105
stopId
transit::Stop, 106
stopLocation
tracking::UserTrackingAlertObject, 153
stops
transit::Route, 90

tracking, 18
tracking::AlertSpecification
inAlertRange, 42
tracking::BusVehicle
alertList, 51
tracking::CommercialTracking
CommercialTracking, 53
getInstance, 53
tracking::GPSLocationObserver
getGPSLocation, 62
gpsUpdate, 62
setGPSLocation, 63
tracking::GPSLocationTracking
registerGPSDevice, 63
unregisterGPSDevice, 63
tracking::GPSPuller
GPSPuller, 65
tracking::GPSPusher
GPSPusher, 66
getInstance, 66
tracking::GPSVehicleTracker
GPSVehicleTracker, 68
tracking::ITrackingService
addUserTrackingAlert, 81
getTransitVehicleLocation, 81
registerVehicleOnRoute, 81
startTrackingController, 82
unregisterVehicleFromRoute, 82
tracking::TrackingAlertFactory
createAlertObserver, 107
tracking::TrackingAlertObserver
getSpec, 108
setSpec, 108
updateAlert, 108
tracking::TrackingDelayAlert
updateAlert, 110
tracking::TrackingLocationAlert
TrackingLocationAlert, 111
tracking::TrackingResponseModel
convertJSONAlertInput, 112
convertJSONVehicleInput, 112
formatJSONResponse, 112
tracking::TrackingServiceController
addUserTrackingAlert, 113
registerVehicleOnRoute, 113
unregisterVehicleFromRoute, 113
tracking::TransitVehicle
addAlertSpecification, 123
checkForAlerts, 123
tracking::TransitVehicleFactory
createTransitVehicle, 123
getGPSTypeFromURL, 123
getVehicleGPSDeviceID, 124
tracking::UserTrackingAlertObject
alertTime, 152
routeID, 152
scheduledTime, 152
stopLocation, 153
transitCoInfo, 153

type, 153
userContactInfo, 153
tracking::VehicleObject
currentRoute, 154
gpsDeviceID, 154
gpsDeviceInfo, 154
transitCoURL, 154
tracking::VehicleRepository
findVehicle, 155
findVehiclesByRoute, 155
removeVehicle, 156
updateVehicle, 156
TrackingAlertFactory, 106
TrackingAlertObserver, 107
TrackingAlertRequestModel, 108
TrackingAlertService, 109
trackingAlertService
alert::service::AlertServiceFactory, 41
TrackingDelayAlert, 110
TrackingLocationAlert, 111
tracking::TrackingLocationAlert, 111
TrackingResponseModel, 111
TrackingServiceController, 112
transit, 19
transit::AbstractFeedParserTemplate
loadFeed, 24
parseFeed, 24
saveRoutes, 24
start, 25
validate, 25
transit::Detour
cause, 55
Detour, 55
detourId, 55
estimatedDelay, 56
transit::Fare
discountedFare, 57
Fare, 56, 57
setDiscountedFare, 57
setRegularFare, 57
transit::GTFSFeedParser
parseFeed, 68
transit::GoogleTransitServiceAdapter
getRoute, 60
getRoutes, 60
GoogleTransitServiceAdapter, 60
transit::ITeamTransitServiceController
alertRequestController, 74
getRoute, 73
getRoutes, 73
getServiceURL, 74
getTransitInfo, 74
handleRouteDisruptionEvent, 74
transitFeed, 74
transit::ITeamTripServiceController
calculateTrip, 75
transit::InvalidRouteParseException
InvalidRouteParseException, 71
routeBatch, 71
serialVersionUID, 71
transit::Location
Location, 83
transit::PersistedTransitFeed
getRoute, 85
getRoutes, 86
transit::Route
detours, 90
Route, 90
routeName, 90
setStops, 90
stops, 90
transit::RouteDisruptionAlert
routeId, 92
transitServiceUrl, 92
transit::RouteDisruptionEvent
RouteDisruptionEvent, 93
transit::RouteRepository
delete, 94
getAll, 94
read, 94
save, 94, 95
transit::RouteSpecification
isSatisfiedBy, 96
transit::Stop
covered, 106
description, 106
getStopTimes, 105
Stop, 105
stopId, 106
transit::TransitFeed
getRoute, 116
getRoutes, 116
transit::TransitInfo
logo, 117
transitAuthorityName, 117
website, 118
transit::TransitProvider
fireRouteDistruptionEvent, 118
name, 119
providerId, 119
registerObserver, 119
unregisterObserver, 119
transit::TransitProviderObserver
handleRouteDisruptionEvent, 120
transit::TransitService
getRoute, 120
getRoutes, 121
getServiceURL, 121
getTransitInfo, 121
transit::Trip
Trip, 125
transit::TripService
calculateTrip, 126
TransitAlertRequestModel, 114
TransitAlertService, 114
transitAlertService

alert::service::AlertServiceFactory, 41
transitAuthorityName
transit::TransitInfo, 117
transitCoInfo
tracking::UserTrackingAlertObject, 153
transitCoURL
tracking::VehicleObject, 154
TransitFeed, 115
transitFeed
transit::ITeamTransitServiceController, 74
TransitInfo, 117
TransitProvider, 118
TransitProviderObserver, 119
TransitService, 120
transitServiceUrl
transit::RouteDisruptionAlert, 92
TransitVehicle, 122
TransitVehicleFactory, 123
Trip, 124
transit::Trip, 125
tripData
alert::domain::model::TripInformation, 126
TripInformation, 125
TripService, 126
type
tracking::UserTrackingAlertObject, 153

UnplannedDisruption
alert::enums::AlertNotificationType, 30
unregisterGPSDevice
tracking::GPSLocationTracking, 63
unregisterObserver
transit::TransitProvider, 119
unregisterVehicleFromRoute
tracking::ITrackingService, 82
tracking::TrackingServiceController, 113
updateAlert
alert::domain::AlertRepository, 33
alert::service::AlertService, 39
alert::service::TrackingAlertService, 109
alert::service::TransitAlertService, 115
alert::service::UserAlertService, 132
tracking::TrackingAlertObserver, 108
tracking::TrackingDelayAlert, 110
updateFavorites
user::UserFavoritesRepository, 134
updateUser
user::ITeamUserManagementService, 80
user::UserManagementService, 146
user::UserRepository, 150
updateVehicle
tracking::VehicleRepository, 156
User, 127
user::User, 128
user, 20
user::FavoriteTransitService
FavoriteTransitService, 58
getFavoriteRouteIds, 58
getTransitServiceUrl, 58
isFavoriteTransitService, 58
setFavoriteRouteIds, 59
setFavoriteTransitService, 59
user::ITeamUserFavoritesService
readFavorites, 76
saveFavorites, 76
user::ITeamUserLoginService
checkPermissions, 77
createAlertSession, 78
getUser, 78
login, 78
logout, 78
resetPassword, 78
sendUsername, 79
user::ITeamUserManagementService
createUser, 80
deleteUser, 80
findUserByEmail, 80
findUserByMobile, 80
findUserByUsername, 80
updateUser, 80
user::Session
getCreationTime, 98
getExpirationTime, 98
getSessionToken, 98
getUserId, 99
isAlertSession, 99
isValid, 99
Session, 98
setExpirationTime, 99
setValid, 99
user::SessionRepository
createSession, 100
getSession, 100
killSession, 101
user::User
getCountryCode, 128
getEmail, 128
getFirstName, 128
getMobile, 129
getPasswordHash, 129
getUserId, 129
getUserType, 129
getUsername, 129
isForcePasswordChange, 129
setCountryCode, 130
setEmail, 130
setFirstName, 130
setForcePasswordChange, 130
setMobile, 130
setPasswordHash, 131
setUserType, 131
User, 128
user::UserFavoritesList
getFavoriteTransitServices, 133
getUserId, 133
setFavoriteTransitServices, 133
UserFavoritesList, 133

user::UserFavoritesRepository
getFavorites, 134
updateFavorites, 134
user::UserFavoritesService
readFavorites, 135
saveFavorites, 136
user::UserLoginService
createAlertSession, 138
getUser, 139
login, 139
logout, 140
resetPassword, 140, 141
sendUsername, 141, 142
user::UserManagementService
createUser, 143
deleteUser, 144
findUserByEmail, 144
findUserByMobile, 145
findUserByUsername, 145
updateUser, 146
user::UserRepository
createUser, 147
deleteUser, 148
getUserByEmail, 148
getUserById, 149
getUserByMobile, 149
getUserByUsername, 150
updateUser, 150
UserAlertRequestModel, 131
UserAlertService, 131
userAlertService
alert::service::AlertServiceFactory, 41
userClient
alert::controller::SessionTokenHandler, 102
alert::service::RouteAlertExecuteStrategy, 91
alert::service::ScheduleAlertExecuteStrategy, 97
userContactInfo
tracking::UserTrackingAlertObject, 153
UserFavoritesList, 133
user::UserFavoritesList, 133
UserFavoritesRepository, 134
UserFavoritesService, 135
UserInformation, 136
userInformation
alert::domain::model::Alert, 27
UserLoginService, 137
UserManagementService, 142
UserRepository, 147
UserSessionInformation, 151
userSessionToken
alert::domain::model::UserSessionInformation, 152
UserTrackingAlertObject, 152
UserType, 153

validate
transit::AbstractFeedParserTemplate, 25
VehicleObject, 154
VehicleRepository, 155
verifySession
alert::controller::AlertRequestController, 36
verifySessionToken
alert::client::AlertUserClient, 44
alert::controller::CertificateHandler, 52
alert::controller::ISessionHandler, 71
alert::controller::SessionTokenHandler, 102

website
transit::TransitInfo, 118