|  |  |
| --- | --- |
|  |  |
|  | wolf_pack |

**Web Services**

December 23, 2016



2170 Georgina Avenue

Santa Monica, CA 90402

800-972-3027

[www.rco.com](http://www.rco.com)

Table of Contents

[1 Introduction 13](#_Toc469976634)

[2 Document History 15](#_Toc469976635)

[3 API Data Structures 26](#_Toc469976636)

[4 Coding Fields 27](#_Toc469976637)

[5 Development 28](#_Toc469976638)

[5.1 Code Organization in Starteam 28](#_Toc469976639)

[5.2 Development Evironments 29](#_Toc469976640)

[5.2.1 Restful Web Services Project Development using eClipse 29](#_Toc469976641)

[5.2.2 Restful Web Services Project Build using Ant. 32](#_Toc469976642)

[5.3 Installing Web Services 33](#_Toc469976643)

[5.3.1 Manual Installation 33](#_Toc469976644)

[5.3.2 Automatic Installation 34](#_Toc469976645)

[5.4 Logging 34](#_Toc469976646)

[5.4.1 How to turn on logging 34](#_Toc469976647)

[5.4.2 Where the logs get stored 34](#_Toc469976648)

[5.5 Test Environment 35](#_Toc469976649)

[5.5.1 Junit Test Cases 35](#_Toc469976650)

[5.5.2 Batch testing 35](#_Toc469976651)

[6 Logistics Web Services 35](#_Toc469976652)

[6.1 How It Works 35](#_Toc469976653)

[6.2 Carriers 35](#_Toc469976654)

[6.3 Restful Web Service 36](#_Toc469976655)

[6.4 Web Service Consumer / Client Program 36](#_Toc469976656)

[6.5 Scheduled Job / Task 37](#_Toc469976657)

[6.6 Installation Procedure 37](#_Toc469976658)

[6.7 Restful Web Service Installation 37](#_Toc469976659)

[6.8 Scheduled Task / Job Configuration 37](#_Toc469976660)

[7 RMS Web Services 38](#_Toc469976661)

[7.1 Alert Service 39](#_Toc469976662)

[7.1.1 createAlertWebFeedback 39](#_Toc469976663)

[7.1.2 getSensorAlerts 39](#_Toc469976664)

[7.1.3 setAlerts 40](#_Toc469976665)

[7.1.4 setSensorAlerts 41](#_Toc469976666)

[7.1.5 getAlerts 42](#_Toc469976667)

[7.1.6 getStoreLowInventoryItems 43](#_Toc469976668)

[7.1.7 getLowInventoryStoreIds 43](#_Toc469976669)

[7.2 Audit Service 44](#_Toc469976670)

[7.2.1 createMobileScanAudit 44](#_Toc469976671)

[7.2.2 getMobileScanAuditIds 45](#_Toc469976672)

[7.3 Billing Service 45](#_Toc469976673)

[7.3.1 createOrganization 45](#_Toc469976674)

[7.3.2 getJobs 48](#_Toc469976675)

[7.3.3 getJobsByDateRangeAndStatus 48](#_Toc469976676)

[7.3.4 createFileInZipcode 49](#_Toc469976677)

[7.3.5 getPhotos 49](#_Toc469976678)

[7.3.6 setPhotos 50](#_Toc469976679)

[7.3.7 setJobPhotos 51](#_Toc469976680)

[7.3.8 setJobs 52](#_Toc469976681)

[7.4 Calendar Service 55](#_Toc469976682)

[7.4.1 getCalendarEvents 55](#_Toc469976683)

[7.4.2 setCalendarEvents 56](#_Toc469976684)

[7.4.3 getEventTypes 59](#_Toc469976685)

[7.4.4 setEventTypes 59](#_Toc469976686)

[7.5 Data Service 60](#_Toc469976687)

[7.5.1 getNestedList 60](#_Toc469976688)

[7.5.2 getFunctionalGroupSyncRecordTypes 61](#_Toc469976689)

[7.5.3 setFunctionalGroupSyncRecordTypes 61](#_Toc469976690)

[7.5.4 getHarvestWeightsFieldsSummary 62](#_Toc469976691)

[7.5.5 getHarvestWeightsUsersDetail 63](#_Toc469976692)

[7.5.6 getHarvestWeightsUsersSummary 63](#_Toc469976693)

[7.5.7 getTemperatures 64](#_Toc469976694)

[7.5.8 getSensorsUpdatedXFiltered 64](#_Toc469976695)

[7.5.9 getWeights 65](#_Toc469976696)

[7.5.10 sendSensorData 65](#_Toc469976697)

[7.5.11 setCompressorSlips 66](#_Toc469976698)

[7.5.12 setContainerTypes 70](#_Toc469976699)

[7.5.13 setFieldLogs 71](#_Toc469976700)

[7.5.14 setFarmOperations 72](#_Toc469976701)

[7.5.15 setFarmTasks 73](#_Toc469976702)

[7.5.16 setFill 75](#_Toc469976703)

[7.5.17 setHarvestAreas 76](#_Toc469976704)

[7.5.18 setHarvestMethods 77](#_Toc469976705)

[7.5.19 setHarvestOperations 78](#_Toc469976706)

[7.5.20 setHarvestSeed 80](#_Toc469976707)

[7.5.21 setHarvestYields 82](#_Toc469976708)

[7.5.22 setMillingSeed 83](#_Toc469976709)

[7.5.23 setPlantings 85](#_Toc469976710)

[7.5.24 setProductCategories 87](#_Toc469976711)

[7.5.25 setProductTypes 89](#_Toc469976712)

[7.5.26 setShippingSeedForms 90](#_Toc469976713)

[7.5.27 setI9LaborForms 92](#_Toc469976714)

[7.5.28 setNoticeToEmployeeForms 96](#_Toc469976715)

[7.5.29 setApplesNorthSliceRoomForms 99](#_Toc469976716)

[7.5.30 setSeedProduction 102](#_Toc469976717)

[7.5.31 setSensorData 104](#_Toc469976718)

[7.5.32 setW4Forms 106](#_Toc469976719)

[7.5.33 setSensors 109](#_Toc469976720)

[7.6 Device Service 110](#_Toc469976721)

[7.6.1 setValves 110](#_Toc469976722)

[7.6.2 setPumps 112](#_Toc469976723)

[7.6.3 setMotors 114](#_Toc469976724)

[7.7 Directory Service 116](#_Toc469976725)

[7.7.1 createRecordInDirectory 116](#_Toc469976726)

[7.7.2 deleteDirectoryBranch 117](#_Toc469976727)

[7.7.3 getAllChildrenList 117](#_Toc469976728)

[7.7.4 getChildrenByTreeId 118](#_Toc469976729)

[7.7.5 getChildrenDirectoryIds 118](#_Toc469976730)

[7.7.6 getDirectoryId 119](#_Toc469976731)

[7.7.7 getDirectoryIds 119](#_Toc469976732)

[7.7.8 getEfileDownloadPath 120](#_Toc469976733)

[7.7.9 getFileFolders 120](#_Toc469976734)

[7.7.10 getFilterCategoryPaths 121](#_Toc469976735)

[7.7.11 getContainerPaths 121](#_Toc469976736)

[7.7.12 getNodeInfoByMobileRecordId 122](#_Toc469976737)

[7.7.13 getNodeInfoByObjectIdObjectType 122](#_Toc469976738)

[7.7.14 getNodeInfoByRecordId 123](#_Toc469976739)

[7.7.15 getNodeInfoByTreeId 123](#_Toc469976740)

[7.7.16 getNodesByItemType 124](#_Toc469976741)

[7.7.17 getNodesByRecordType 124](#_Toc469976742)

[7.7.18 getChildNodes 124](#_Toc469976743)

[7.7.19 getEncryptMode 125](#_Toc469976744)

[7.7.20 readWebFile 125](#_Toc469976745)

[7.7.21 setDirectoryNodeName 126](#_Toc469976746)

[7.7.22 writeWebFile 126](#_Toc469976747)

[7.8 Email Service 127](#_Toc469976748)

[7.8.1 EmailFileToList 127](#_Toc469976749)

[7.8.2 EmailMessageToFunctionalGroup 128](#_Toc469976750)

[7.8.3 SendEmail 128](#_Toc469976751)

[7.9 Forms Service 129](#_Toc469976752)

[7.9.1 setBoxes 129](#_Toc469976753)

[7.9.2 setCentralFiles 130](#_Toc469976754)

[7.9.3 setEngineeringDrawings 132](#_Toc469976755)

[7.9.4 setLibraryMaterials 133](#_Toc469976756)

[7.9.5 setRecordRequests 135](#_Toc469976757)

[7.9.6 setSignatures 136](#_Toc469976758)

[7.10 MapService 137](#_Toc469976759)

[7.10.1 getZipcodePlus4 137](#_Toc469976760)

[7.10.2 getGpsCoordinatesFromAddress 137](#_Toc469976761)

[7.10.3 getOptimalRoute 138](#_Toc469976762)

[7.10.4 setFields 139](#_Toc469976763)

[7.10.5 setLocations 141](#_Toc469976764)

[7.10.6 setRanches 142](#_Toc469976765)

[7.11 Mobile Service 144](#_Toc469976766)

[7.11.1 getSensorData 144](#_Toc469976767)

[7.11.2 uploadFolderFile 145](#_Toc469976768)

[7.11.3 uploadScanData 145](#_Toc469976769)

[7.11.4 uploadScanFile 147](#_Toc469976770)

[7.11.5 uploadScanLog 148](#_Toc469976771)

[7.11.6 createVisitor 149](#_Toc469976772)

[7.11.7 getDeviceId 150](#_Toc469976773)

[7.11.8 setSalesTypes 150](#_Toc469976774)

[7.11.9 setSalesRecords 151](#_Toc469976775)

[7.12 Part Service 153](#_Toc469976776)

[7.12.1 changeItemNumber 153](#_Toc469976777)

[7.12.2 createNonTrackedPart 154](#_Toc469976778)

[7.12.3 createPart 154](#_Toc469976779)

[7.12.4 createParts 155](#_Toc469976780)

[7.12.5 createPart 156](#_Toc469976781)

[7.12.6 createPartWithCoding 156](#_Toc469976782)

[7.12.7 createTrackedPart 157](#_Toc469976783)

[7.12.8 createStore 157](#_Toc469976784)

[7.12.9 findTruck 158](#_Toc469976785)

[7.12.10 getAllItemLocationsByStore 158](#_Toc469976786)

[7.12.11 getCycleCount 158](#_Toc469976787)

[7.12.12 getInventoryCounts 159](#_Toc469976788)

[7.12.13 getItemIdInfo 159](#_Toc469976789)

[7.12.14 getPacking 159](#_Toc469976790)

[7.12.15 getPartIdInfo 160](#_Toc469976791)

[7.12.16 getPendingServiceTickets 160](#_Toc469976792)

[7.12.17 getStoreHeaderId 160](#_Toc469976793)

[7.12.18 getStoreNumberForLocationRecordId 161](#_Toc469976794)

[7.12.19 getStoreInventoryList 161](#_Toc469976795)

[7.12.20 getStorePartCount 162](#_Toc469976796)

[7.12.21 getStorePartIdsAll 162](#_Toc469976797)

[7.12.22 getStorePartIdsUpdated 162](#_Toc469976798)

[7.12.23 getTruckPacking 163](#_Toc469976799)

[7.12.24 movePart 163](#_Toc469976800)

[7.12.25 moveParts 164](#_Toc469976801)

[7.12.26 movePartsOffline 165](#_Toc469976802)

[7.12.27 receivePartHeader 166](#_Toc469976803)

[7.12.28 receivePartDetail 169](#_Toc469976804)

[7.12.29 PartsImport 171](#_Toc469976805)

[7.12.30 recodeTrackedItem 171](#_Toc469976806)

[7.12.31 setAdjustments 171](#_Toc469976807)

[7.12.32 setChangeItemNumbers 172](#_Toc469976808)

[7.12.33 setCows 173](#_Toc469976809)

[7.12.34 setCycleCounts 173](#_Toc469976810)

[7.12.35 setErrorMessage 175](#_Toc469976811)

[7.12.36 setMailPackages 176](#_Toc469976812)

[7.12.37 setMilkingHistories 178](#_Toc469976813)

[7.12.38 setMovePartsViolations 180](#_Toc469976814)

[7.12.39 setPacking 182](#_Toc469976815)

[7.12.40 setParts 185](#_Toc469976816)

[7.12.41 setQuickBooksOnlineLog 188](#_Toc469976817)

[7.12.42 setPartsImport 189](#_Toc469976818)

[7.12.43 setReceiveParts 192](#_Toc469976819)

[7.12.44 setReceivePartsSingle 194](#_Toc469976820)

[7.12.45 setFieldBoxes 195](#_Toc469976821)

[7.12.46 fillProducePacks 196](#_Toc469976822)

[7.12.47 setForklifts 197](#_Toc469976823)

[7.12.48 setHarvestWeight 198](#_Toc469976824)

[7.12.49 setHarvestWeights 199](#_Toc469976825)

[7.12.50 setPalletJacks 200](#_Toc469976826)

[7.12.51 setServiceTickets 201](#_Toc469976827)

[7.12.52 setServiceTicketsSingle 202](#_Toc469976828)

[7.12.53 setShippingBoxes 203](#_Toc469976829)

[7.12.54 setShipParts 204](#_Toc469976830)

[7.12.55 setShipPartsNoReturnData 206](#_Toc469976831)

[7.12.56 setShipPartsSingle 208](#_Toc469976832)

[7.12.57 setShips 208](#_Toc469976833)

[7.12.58 setStores 210](#_Toc469976834)

[7.12.59 setSyncMessage 212](#_Toc469976835)

[7.12.60 setTaxis 212](#_Toc469976836)

[7.12.61 setTemperatures 214](#_Toc469976837)

[7.12.62 setTrailers 216](#_Toc469976838)

[7.12.63 setTrucks 219](#_Toc469976839)

[7.12.64 setWeights 220](#_Toc469976840)

[7.12.65 setupPart 222](#_Toc469976841)

[7.12.66 shipPart 222](#_Toc469976842)

[7.12.67 updateSerialNumbers 224](#_Toc469976843)

[7.13 Record Service 225](#_Toc469976844)

[7.13.1 appendRecordContent 225](#_Toc469976845)

[7.13.2 checkinRecord 225](#_Toc469976846)

[7.13.3 checkoutRecord 226](#_Toc469976847)

[7.13.4 copyUserRecord 226](#_Toc469976848)

[7.13.5 createDocumentRecord 227](#_Toc469976849)

[7.13.6 createService 227](#_Toc469976850)

[7.13.7 createRecordInDateBranch 228](#_Toc469976851)

[7.13.8 deleteRecord 228](#_Toc469976852)

[7.13.9 deleteRecords 228](#_Toc469976853)

[7.13.10 deleteRecordsByRecordTypeAndUser 229](#_Toc469976854)

[7.13.11 emailLoginInfo 229](#_Toc469976855)

[7.13.12 getCsvContent 230](#_Toc469976856)

[7.13.13 getLatLonByRecordType 230](#_Toc469976857)

[7.13.14 getLibraryCoding 231](#_Toc469976858)

[7.13.15 getLibraryItemsByRecordType 231](#_Toc469976859)

[7.13.16 getRecordCoding 231](#_Toc469976860)

[7.13.17 getRecordCodingByRecordId 232](#_Toc469976861)

[7.13.18 getRecordCodingByMobileRecordId 232](#_Toc469976862)

[7.13.19 getRecordCount 232](#_Toc469976863)

[7.13.20 getRecordContent 233](#_Toc469976864)

[7.13.21 getRecordContentAsByteData 233](#_Toc469976865)

[7.13.22 getRecordContentUnencrypted 234](#_Toc469976866)

[7.13.23 getRecordFilterCodingFields 234](#_Toc469976867)

[7.13.24 getRecordIdsAll 234](#_Toc469976868)

[7.13.25 getRecordIdsUpdated 235](#_Toc469976869)

[7.13.26 getRecordIdsAllByRecordType 235](#_Toc469976870)

[7.13.27 getRecordIdsUpdatedByRecordType 236](#_Toc469976871)

[7.13.28 getRecordsUpdated 236](#_Toc469976872)

[7.13.29 getRecordsUpdatedFiltered 237](#_Toc469976873)

[7.13.30 getRecordsUpdatedX 238](#_Toc469976874)

[7.13.31 getRecordsUpdatedXFiltered 238](#_Toc469976875)

[7.13.32 getRecordInfo 240](#_Toc469976876)

[7.13.33 getRecordTypesByGroup 240](#_Toc469976877)

[7.13.34 getTemplatePartIds 241](#_Toc469976878)

[7.13.35 movePhysicalRecord 241](#_Toc469976879)

[7.13.36 setRecordsByRecordTypeAndUser 241](#_Toc469976880)

[7.13.37 setRecordCodingTimestamp 243](#_Toc469976881)

[7.13.38 setRecordContent 243](#_Toc469976882)

[7.13.39 setRecordContentFile 244](#_Toc469976883)

[7.13.40 setRecordCoding 244](#_Toc469976884)

[7.13.41 setRecordCodingByRecordId 244](#_Toc469976885)

[7.13.42 setRecordContentTimestamp 245](#_Toc469976886)

[7.13.43 setRecordName 245](#_Toc469976887)

[7.13.44 setRecordCodingFields 246](#_Toc469976888)

[7.13.45 setServiceRequests 248](#_Toc469976889)

[7.14 Report Service 249](#_Toc469976890)

[7.14.1 getInvoiceChartData 249](#_Toc469976891)

[7.14.2 getTimecardChartData 249](#_Toc469976892)

[7.15 Quickbook Service 250](#_Toc469976893)

[7.15.1 createChartHeader 250](#_Toc469976894)

[7.15.2 createInvoiceHeader 251](#_Toc469976895)

[7.15.3 createInvoiceDetails 252](#_Toc469976896)

[7.15.4 createItemReceiptHeader 253](#_Toc469976897)

[7.15.5 createItemReceiptDetails 255](#_Toc469976898)

[7.15.6 createPayment 256](#_Toc469976899)

[7.15.7 createPurchaseOrderHeader 256](#_Toc469976900)

[7.15.8 createPurchaseOrderDetails 257](#_Toc469976901)

[7.15.9 createRefund 259](#_Toc469976902)

[7.15.10 createRMADetail 259](#_Toc469976903)

[7.15.11 createReportHeader 260](#_Toc469976904)

[7.15.12 createRMAHeader 261](#_Toc469976905)

[7.15.13 createVoid 261](#_Toc469976906)

[7.15.14 getAccounts 262](#_Toc469976907)

[7.15.15 getBills 262](#_Toc469976908)

[7.15.16 getCustomerRmaDetailIds 262](#_Toc469976909)

[7.15.17 getCustomerRmaHeaderId 263](#_Toc469976910)

[7.15.18 getCustomerRMAs 263](#_Toc469976911)

[7.15.19 getItemReceiptDetailIds 264](#_Toc469976912)

[7.15.20 getChartHeaderIdsUpdated 264](#_Toc469976913)

[7.15.21 getChartHeaderIdsAll 264](#_Toc469976914)

[7.15.22 getContracts 265](#_Toc469976915)

[7.15.23 getDealers 265](#_Toc469976916)

[7.15.24 getExpenses 266](#_Toc469976917)

[7.15.25 getInvoices 266](#_Toc469976918)

[7.15.26 getInvoicesByDateAndSalesRep 266](#_Toc469976919)

[7.15.27 getInvoiceHeaderId 267](#_Toc469976920)

[7.15.28 getInvoiceDetailIds 267](#_Toc469976921)

[7.15.29 getMatters 268](#_Toc469976922)

[7.15.30 getNewInvoiceHeaderIds 268](#_Toc469976923)

[7.15.31 getNewItemReceiptHeaderIds 268](#_Toc469976924)

[7.15.32 getNewPaymentHeaderIds 269](#_Toc469976925)

[7.15.33 getOpenInvoices 269](#_Toc469976926)

[7.15.34 getOpenPurchaseOrders 269](#_Toc469976927)

[7.15.35 getPayment 270](#_Toc469976928)

[7.15.36 getPaymentByMobileInvoiceNumber 270](#_Toc469976929)

[7.15.37 getPickList 271](#_Toc469976930)

[7.15.38 getPurchaseOrderDetailIds 271](#_Toc469976931)

[7.15.39 getPurchaseOrderHeaderId 271](#_Toc469976932)

[7.15.40 getPurchaseOrders 272](#_Toc469976933)

[7.15.41 getQuantityOnHand 272](#_Toc469976934)

[7.15.42 getRefund 273](#_Toc469976935)

[7.15.43 getRefundDetailIds 273](#_Toc469976936)

[7.15.44 getRefunds 274](#_Toc469976937)

[7.15.45 getRefundsByTimestamp 274](#_Toc469976938)

[7.15.46 getReportHeaderIdsAll 274](#_Toc469976939)

[7.15.47 getReportHeaderIdsUpdated 275](#_Toc469976940)

[7.15.48 getSalesOrders 275](#_Toc469976941)

[7.15.49 getSalesTaxCodes 275](#_Toc469976942)

[7.15.50 getSalesTaxRates 276](#_Toc469976943)

[7.15.51 getServices 276](#_Toc469976944)

[7.15.52 getTerms 277](#_Toc469976945)

[7.15.53 getTransactions 277](#_Toc469976946)

[7.15.54 getVendorRmaDetailIds 278](#_Toc469976947)

[7.15.55 getVendorRmaHeaderId 278](#_Toc469976948)

[7.15.56 getVendorRMAs 278](#_Toc469976949)

[7.15.57 getVoid 279](#_Toc469976950)

[7.15.58 incrementInvoiceDetailQtyPicked 279](#_Toc469976951)

[7.15.59 setAccounts 280](#_Toc469976952)

[7.15.60 setBills 281](#_Toc469976953)

[7.15.61 setDealers 284](#_Toc469976954)

[7.15.62 setExpenses 285](#_Toc469976955)

[7.15.63 setInvoices 286](#_Toc469976956)

[7.15.64 setMatters 292](#_Toc469976957)

[7.15.65 setPayments 292](#_Toc469976958)

[7.15.66 setPurchaseOrders 294](#_Toc469976959)

[7.15.67 setCustomerRMAs 296](#_Toc469976960)

[7.15.68 setCustomerRMAsNoReturnData 300](#_Toc469976961)

[7.15.69 setPriceCodes 302](#_Toc469976962)

[7.15.70 setPriceLevels 304](#_Toc469976963)

[7.15.71 setRecordProcessed 305](#_Toc469976964)

[7.15.72 setRefunds 306](#_Toc469976965)

[7.15.73 setSalesOrders 310](#_Toc469976966)

[7.15.74 setSalesTaxCodes 312](#_Toc469976967)

[7.15.75 setSalesTaxRates 314](#_Toc469976968)

[7.15.76 setServices 315](#_Toc469976969)

[7.15.77 setTerms 316](#_Toc469976970)

[7.15.78 setVendorRMAs 317](#_Toc469976971)

[7.16 Sales Service 321](#_Toc469976972)

[7.16.1 setProspects 321](#_Toc469976973)

[7.17 Security Services 323](#_Toc469976974)

[7.17.1 getAssignedRoleNames 323](#_Toc469976975)

[7.17.2 getUserInfo 323](#_Toc469976976)

[7.17.3 getUserRights 324](#_Toc469976977)

[7.17.4 isAssignedRight 324](#_Toc469976978)

[7.17.5 isAssignedRole 324](#_Toc469976979)

[7.17.6 isLoginUnique 325](#_Toc469976980)

[7.17.7 remoteSessionLogin 325](#_Toc469976981)

[7.18 Ship Services 325](#_Toc469976982)

[7.18.1 createBolHeader 325](#_Toc469976983)

[7.18.2 createBillOfLadingDetail 326](#_Toc469976984)

[7.18.3 createBillOfLadingSignature 327](#_Toc469976985)

[7.18.4 createShipHeader 327](#_Toc469976986)

[7.18.5 createShipDetail 328](#_Toc469976987)

[7.18.6 createShipSignature 328](#_Toc469976988)

[7.18.7 getBillOfLadingDetailIds 329](#_Toc469976989)

[7.18.8 getBillOfLadingHeaderId 329](#_Toc469976990)

[7.18.9 getBillOfLadingHeaderIds 329](#_Toc469976991)

[7.18.10 getBillOfLadingSignatureIds 330](#_Toc469976992)

[7.18.11 getCityLocator 330](#_Toc469976993)

[7.18.12 getHarvestCalendar 331](#_Toc469976994)

[7.18.13 getEngineData 332](#_Toc469976995)

[7.18.14 getManifest 332](#_Toc469976996)

[7.18.15 getNewShipHeaderIds 333](#_Toc469976997)

[7.18.16 getShipDetailIds 333](#_Toc469976998)

[7.18.17 getShipHeaderId 334](#_Toc469976999)

[7.18.18 getShipSignatureIds 334](#_Toc469977000)

[7.18.19 getShipTrackStatus 334](#_Toc469977001)

[7.18.20 getShipWaybills 335](#_Toc469977002)

[7.18.21 getTimeAndDistance 335](#_Toc469977003)

[7.18.22 getTrackingInfo 335](#_Toc469977004)

[7.18.23 getTrailerData 336](#_Toc469977005)

[7.18.24 getTruckAlerts 336](#_Toc469977006)

[7.18.25 getTruckEngineDetails 337](#_Toc469977007)

[7.18.26 getTruckEngineNow 337](#_Toc469977008)

[7.18.27 getTruckGPSNow 337](#_Toc469977009)

[7.18.28 getTruckInfo 338](#_Toc469977010)

[7.18.29 getTruckLocationsFiltered 338](#_Toc469977011)

[7.18.30 getTruckLogDetails 339](#_Toc469977012)

[7.18.31 getTruckTemperatureDetails 339](#_Toc469977013)

[7.18.32 getTruckSensorsNow 340](#_Toc469977014)

[7.18.33 getTrucksInfo 340](#_Toc469977015)

[7.18.34 getTrucksInGeofences 340](#_Toc469977016)

[7.18.35 getUpsTrackNumberInfo 341](#_Toc469977017)

[7.18.36 setBillOfLadings 341](#_Toc469977018)

[7.18.37 setBOLTruckRoute 344](#_Toc469977019)

[7.18.38 setShipWaybills 344](#_Toc469977020)

[7.18.39 getShipWaybillsNotProcessed 345](#_Toc469977021)

[7.18.40 setCityLocators 346](#_Toc469977022)

[7.18.41 setMailHistory 346](#_Toc469977023)

[7.18.42 setMailLocations 348](#_Toc469977024)

[7.18.43 setPickLists 350](#_Toc469977025)

[7.18.44 setShipLogs 352](#_Toc469977026)

[7.18.45 setTrailerLoadings 354](#_Toc469977027)

[7.18.46 setTruckAlerts 357](#_Toc469977028)

[7.18.47 setTruckDVIRS 358](#_Toc469977029)

[7.18.48 setTruckELDs 363](#_Toc469977030)

[7.18.49 setTruckEngines 366](#_Toc469977031)

[7.18.50 setTruckGeofences 368](#_Toc469977032)

[7.18.51 setTruckLogs 369](#_Toc469977033)

[7.18.52 setTruckRoutes 373](#_Toc469977034)

[7.18.53 setTruckRules 375](#_Toc469977035)

[7.18.54 setTruckSensors 377](#_Toc469977036)

[7.18.55 setTruckTrips 378](#_Toc469977037)

[7.18.56 setTruckDriverViolations 380](#_Toc469977038)

[7.19 SMS Service 382](#_Toc469977039)

[7.19.1 SmsMessageToFunctionalGroup 382](#_Toc469977040)

[7.20 System Service 382](#_Toc469977041)

[7.20.1 setWebsiteMessage 382](#_Toc469977042)

[7.20.2 getServerName 383](#_Toc469977043)

[7.20.3 getSystemStatus 383](#_Toc469977044)

[7.21 Timecard Service 383](#_Toc469977045)

[7.21.1 createTimecardHeader 384](#_Toc469977046)

[7.21.2 createTimecardTask 384](#_Toc469977047)

[7.21.3 createTimecardPart 384](#_Toc469977048)

[7.21.4 createTimecardPhoto 385](#_Toc469977049)

[7.21.5 getTimecardsFiltered 385](#_Toc469977050)

[7.21.6 getTimecardsApproved 386](#_Toc469977051)

[7.21.7 getTimecardsForInvoicing 386](#_Toc469977052)

[7.21.8 getTimecardHeaderIds 387](#_Toc469977053)

[7.21.9 getTimecardHeaderIdsByDateRange 388](#_Toc469977054)

[7.21.10 getTimecardHeaderIdsByStatus 388](#_Toc469977055)

[7.21.11 getTimecardTotalHours 389](#_Toc469977056)

[7.21.12 getTimecardSubmitters 389](#_Toc469977057)

[7.21.13 getTimecardTaskIds 390](#_Toc469977058)

[7.21.14 getTimecardPartIds 390](#_Toc469977059)

[7.21.15 getTimecardNonBillableHours 391](#_Toc469977060)

[7.21.16 getTimecardYearToDateHours 391](#_Toc469977061)

[7.21.17 getUserTimecardsByStatus 391](#_Toc469977062)

[7.21.18 getUserTimecardsByStatusAndLogin 392](#_Toc469977063)

[7.21.19 setTimecardDetailAsCustomerBilled 393](#_Toc469977064)

[7.21.20 setTimecardDetailAsVendorBilled 393](#_Toc469977065)

[7.21.21 setTimecards 393](#_Toc469977066)

[7.21.22 setTimecardStatus 397](#_Toc469977067)

[7.22 UserService 398](#_Toc469977068)

[7.22.1 authenticateUser 398](#_Toc469977069)

[7.22.2 createAlertUser 398](#_Toc469977070)

[7.22.3 createUser 399](#_Toc469977071)

[7.22.4 createSecurityUser 399](#_Toc469977072)

[7.22.5 createUserGroup 400](#_Toc469977073)

[7.22.6 deleteAlertUser 401](#_Toc469977074)

[7.22.7 deleteSecurityUser 401](#_Toc469977075)

[7.22.8 deleteUser 401](#_Toc469977076)

[7.22.9 isUserLoginAvailable 402](#_Toc469977077)

[7.22.10 getCustomerAlertInfo 402](#_Toc469977078)

[7.22.11 getFunctionalGroupMembers 403](#_Toc469977079)

[7.22.12 getFunctionalGroupMaps 403](#_Toc469977080)

[7.22.13 getFunctionalGoupAlertInfo 403](#_Toc469977081)

[7.22.14 getUserCoding 404](#_Toc469977082)

[7.22.15 getUserGroupMaps 404](#_Toc469977083)

[7.22.16 getUserId 404](#_Toc469977084)

[7.22.17 getUsers 405](#_Toc469977085)

[7.22.18 getUserInfo 405](#_Toc469977086)

[7.22.19 moveAlertUser 406](#_Toc469977087)

[7.22.20 moveSecurityUser 406](#_Toc469977088)

[7.22.21 moveUser 407](#_Toc469977089)

[7.22.22 moveUserToNamedGroup 408](#_Toc469977090)

[7.22.23 renameUserGroup 408](#_Toc469977091)

[7.22.24 sendUserPassword 409](#_Toc469977092)

[7.22.25 setLaborGroups 409](#_Toc469977093)

[7.22.26 setUserLogin 411](#_Toc469977094)

[7.22.27 setUserPassword 411](#_Toc469977095)

[7.22.28 setUserRole 412](#_Toc469977096)

[7.22.29 setUserShipTos 412](#_Toc469977097)

[7.22.30 setUsers 414](#_Toc469977098)

[7.22.31 updateAlertUser 415](#_Toc469977099)

[7.22.32 updateSecurityUser 416](#_Toc469977100)

# Introduction

This document describes the billing system which generates monthly bills for all the clients of a given organization. In the following figure you can see the 3 main modules. There is a local computer which is a window machine that contains the accounting software (Intuit – Quickbooks). You have the RMS Server which RCO maintains which acts as the message hub. You have a mobile device where the operator enters the time card information. The worker can also use an internet browser to enter time card information.

An organization has the workers (employees and contractors) submit time cards which contains one or more tasks. A task can contain one or more parts and photos. A few days after the end of each month or if manually requested a workflow process is run on the local machine that creates invoices for all forms that have not currently been billed. These invoices are then emailed to the client using the default email system. Note that some clients may have no invoices while other clients may have several invoices. When a contractor submits a timecard a client invoice gets generated and also a contractor bill gets generated.



Figure 1.0 System Block Diagram

The workers communicate to the RMS Server via web browser or their iphone using a secure socket layer internet connection.

**Notes**

In all the calls below you replace the string {webserver} with something like {https://www.ServerName.com/Image2000/rest.

**RETURNS:**

**All the web services return the same status codes. It is best to examing the tomcat stdout logs for better information.**

|  |  |  |
| --- | --- | --- |
| **Status Code** | **Name** | **Notes** |
| 200 | Successful |  |
| 400 | Bad Request | [rfc2616#sec10.4.1](http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1), returned if request not understood, see Warning: response header |
| 403 | Forbidden |  |
| 404 | Not Found |  |

**RESPONSE:**

|  |  |  |
| --- | --- | --- |
| **Response-Header** | **Name** | **Notes** |
| Content-Type | Returned with Entities | http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.17 |
| Date | Always returned | http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.18 |
| ETag | Returned for 200 status | http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.19 |
| Last-Modified | Returned for 200 status | http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.29 |

When a CSV file is passed as part of the json POST method then you enclose each data element in quotes and there is a comma that seperates each element. At the end of each line is the CRLF symbols.

# Document History

The following table shows the document version history change to the document named ws.doc and stored in the folder \\bear\rms\doc\billing\design\spec.

|  |  |  |
| --- | --- | --- |
| ***Author*** | ***Date*** | ***Revision Description*** |
| Roy Nabel | 3/9/2011 | Creation |
| Roy Nabel | 3/15/2011 | Major edits to many sections |
| Roy Nabel | 3/17/2011 | Qb, api, iphone screens |
| Roy Nabel | 3/20/2011 | Iphone mode buttons, removed menu |
| Roy Nabel | 3/21/2011 | Major iphone screen mods |
| Roy Nabel | 3/22/2011 | Major iphone screen mods & web svcs |
| Roy Nabel | 3/25/2011 | Task & client api |
| Roy Nabel | 3/26/2011 | client api |
| Roy Nabel | 3/27/2011 | Timecard api, iphone setup & job detail |
| Roy Nabel | 4/4/2011 | Api, stock |
| Roy Nabel | 4/6/2011 | Add worker Api, qb email, qb worker |
| Roy Nabel | 4/9/2011 | Added directory api |
| Roy Nabel | 4/10/2011 | Directory api; added rms content |
| Roy Nabel | 4/19/2011 | Modified custom fields |
| Roy Nabel | 4/20/2011 | Modified QB synchronize |
| Roy Nabel | 4/21/2011 | Add iphone timer, api, sync |
| Roy Nabel | 4/22/2011 | Added api getdirectoryobjectid & qb menu item check data |
| Roy Nabel | 4/23/2011 | Added security api; updated check data; added new custom fields to match sync program key.csv file for qb customers. |
| Roy Nabel | 4/25/2011 | Modified Directory api; added directory tests; add qb check data dialog. |
| Roy Nabel | 4/27/2011 | Modified er diagram; directory & user api |
| Roy Nabel | 4/28/2011 | Modified directory, services & user api |
| Roy Nabel | 4/29/2011 | Modified rms time card form |
| Roy Nabel | 5/1/2011 | Modified rms time card form |
| Don Zhang | 5/3/2011 | Add restful json api |
| Roy Nabel | 5/4/2011 | Added to rms timecard form |
| Roy Nabel | 5/5/2011 | Modified qb, directory, iphone api, er diagram |
| Roy Nabel | 5/7/2011 | Modified qb, iphone api |
| Don Zhang | 5/9/2011 | Web services |
| Roy Nabel | 5/17/2011 | Added deleteRecord api |
| Roy Nabel | 5/24/2011 | Modified createPart, createService api |
| Roy Nabel | 5/25/2011 | Modified syncStoresToInventory |
| Roy Nabel | 5/27/2011 | Replaced login/password with sessionId |
| Roy Nabel | 5/30/2011 | Added web services: getPartIds, createVisitor |
| Roy Nabel | 6/2/2011 | Added uploadScanData in mobile services |
| Roy Nabel | 6/14/2011 | Added getStorePartIdsAll & getStorePartIdsUpdated, modified getRecordIdsAll to include store |
| Roy Nabel | 6/15/2011 | Added custom fields for item list in quick books section |
| Roy Nabel | 6/19/2011 | Added stores; modified clients, settings |
| Roy Nabel | 6/22/2011 | Add logging section; added uploadScanFile |
| Roy Nabel | 6/28/2011 | Add timecard web services & rms functions |
| Roy Nabel | 6/29/2011 | Modified rms timecard arguments; added linking diagrm |
| Roy Nabel | 6/30/2011 | Modified introduction; timecard api changes |
| Roy Nabel | 7/2/2011 | Iphone sync, map+, call+, msg+, timecard svcs |
| Roy Nabel | 7/4/2011 | timecardHeaderId; iphone settings, iphone login, iphone outline changes |
| Roy Nabel | 7/5/2011 | Added web service api’s to handle iphone login forgot login, forgot password. |
| Roy Nabel | 7/7/2011 | Removed some timecard web services |
| Roy Nabel | 7/9/2011 | Worked on timecard service examples. |
| Roy Nabel | 7/12/2011 | Added audit section and 2 functions |
| Roy Nabel | 7/17/2011 | Added audit, rma web services |
| Roy Nabel | 7/19/2011 | Added several quick books web services |
| Roy Nabel | 7/20/2011 | Added getTemplatePartIds in record web services. |
| Roy Nabel | 7/23/2011 | Modified rma, invoice, item receipt and purchase order web service header functions |
| Roy Nabel | 7/26/2011 | Added serveral new part web services |
| Roy Nabel | 7/27/2011 | Deleted kit, added new part web services |
| Roy Nabel | 7/28/2011 | Moved rma to quickbooks, add web service |
| Roy Nabel | 7/29/2011 | Updated receive web service processing |
| Roy Nabel | 7/30/2011 | Added rma web services, worked on iphone |
| Roy Nabel | 7/31/2011 | Modified move & receive part web svcs |
| Roy Nabel | 8/1/2011 | Modified trackUpsId, added create store; modified iphone home list |
| Roy Nabel | 8/2/2011 | Added getPartInfo, createStore, getStoreHeaderId |
| Roy Nabel | 8/4/2011 | Modified receivePart & shipPart |
| Roy Nabel | 8/6/2011 | receivePart, shipPart, createPurchaseOrderDetail |
| Roy Nabel | 8/7/2011 | Modified createInvoiceHeader, shipPart, iphone home screen |
| Roy Nabel | 8/15/2011 | Added receivePart; getLocalAdminUser; getUpsTrackingHeaderIds |
| Roy Nabel | 8/16/2011 | Added several new quickbook services to handle charts, payments and reports |
| Roy Nabel | 8/18/2011 | Added figures & text to chartheader and reportheader |
| Roy Nabel | 8/19/2011 | Tested all the new web svcs except receiveScannedParts |
| Roy Nabel | 9/8/2011 | Modified getTimecardHeaderIds |
| Roy Nabel | 9/17/2011 | Added createRecord and createPdfEfileRecord |
| Roy Nabel | 9/19/2011 | changed createPdfEfileRecord to createDocumentRecord |
| Randy Nichols | 9/23/2011 | Add new directory service getNodesByItemType for print2Pdf |
| Roy Nabel | 11/6/2011 | Added new ship api’s |
| Don Zhang | 11/8/2011 | Added carrier section |
| Roy Nabel | 11/9/2011 | Added new security getUserRole |
| Roy Nabel | 11/14/2011 | Extensive modifications to ship services |
| Don Zhang | 11/24/2011 | Added setRecordCodingFields |
| Roy Nabel | 12/8/2011 | Added getStoreInventoryList |
| Roy Nabel | 12/21/2011 | Added checkinRecords, checkoutRecords |
| Roy Nabel | 1/4/2012 | modified checkinRecord, checkoutRecord |
| Roy Nabel | 1/9/2012 | modified checkinRecord, checkoutRecord |
| Roy Nabel | 1/12/2012 | Removed redundant material. |
| Roy Nabel | 1/17/2012 | Added coding fields section |
| Roy Nabel | 1/19/2012 | Removed all calls location with LocationRecordId |
| Roy Nabel | 1/23/2012 | Removed receiveScanParts; added getRecordIsAllByrecordType; added getRecordIdsUpdatedByRecordType |
| Roy Nabel | 2/1/2012 | Added setTimecardStatus |
| Roy Nabel | 2/10/2012 | Added getOpenInvoices, getPickList |
| Roy Nabel | 2/13/2012 | Added incrementInvoiceDetailQtyPicked |
| Roy Nabel | 2/13/2012 | Added incrementInvoiceDetailQtyed |
| Roy Nabel | 3/14/2012 | Added getNestedList |
| Roy Nabel | 3/17/2012 | Added createUserGroup, renameUserGroup |
| Roy Nabel | 3/19/2012 | Added moveUser; setUserRole |
| Roy Nabel | 3/20/2012 | Added setUserPassword; getUserId |
| Roy Nabel | 3/21/2012 | Added role to createUser |
| Roy Nabel | 4/23/2012 | Added getRecordTypes |
| Roy Nabel | 5/2/2012 | Added getRecordFilters |
| Roy Nabel | 5/8/2012 | Added getAccounts, getServices, getTaxRates, getTaxCodes, getUsers, setAccounts, setServices, setTaxRates, setTaxCodes, setUsers, getTimecardsApproved, GetTimecardsForInvoicing |
| Roy Nabel | 5/10/2012 | Modified creatinvoiceheader and createinvoicedetail to take csv file for arguments to speed up quickbooks service. Added mobileInvoiceNumber argument to createinvoiceheader. |
| Roy Nabel | 5/11/2012 | getTimecardsApproved & getTimecardsForInvoicing |
| Roy Nabel | 5/14/2012 | Added getDeviceId |
| Roy Nabel | 5/17/2012 | Modified setUsers; setTaxRates, setTaxCodes |
| Roy Nabel | 5/19/2012 | Added setInvoices, getInvoices |
| Roy Nabel | 5/23/2012 | Modified setInvoices, createInvoiceHeader, createInvoiceDetail |
| Roy Nabel | 5/24/2012 | Modified setInvoices; Added getUserCoding, setRecordContentFile |
| Roy Nabel | 5/29/2012 | Added getUserFiles, getRecordPropertiesFile |
| Roy Nabel | 6/4/2012 | Removed getUserFiles, getRecordPropertiesFile; changed | to CRLF |
| Roy Nabel | 6/5/2012 | Added getOpenPurchaseOrders, getPurchaseOrders, setPurchaseOrders, getTerms, setTerms |
| Roy Nabel | 6/9/2012 | Updated setPurchaseOrders file arguments |
| Roy Nabel | 6/14/2012 | Added setTimecards, getTimecardsFiltered, getRecordContentUnencrypted |
| Roy Nabel | 6/25/2012 | Modified setInvoices with mobileXXX coding fields. |
| Roy Nabel | 7/3/2012 | Added getFunctionalGroupSyncList and setFunctionalGroupSyncList |
| Roy Nabel | 7/6/2012 | Change name getFunctionalGroupSyncRecordTypes and setFunctionalGroupSyncRecordTypes; added createPayment, createRefund, createVoid, getPayment, getRefund, getVoid |
| Roy Nabel | 7/29/2012 | Added deleteUser |
| Roy Nabel | 8/6/2012 | Added getPaymentByMobileInvoiceNumber; removed createRecordInStorageBranch |
| Roy Nabel | 8/7/2012 | Added getUserTimecards |
| Roy Nabel | 8/9/2012 | Added getTransactions |
| Roy Nabel | 8/15/2012 | Added setRefunds, getRefunds |
| Roy Nabel | 8/27/2012 | Added getRefundDetailIds |
| Roy Nabel | 8/28/2012 | Modified setTimecards; getTimecardsForInvoicing |
| Roy Nabel | 8/31/2012 | Add moveUserToNamedGroup |
| Roy Nabel | 9/5/2012 | Modified setInvoices renamed SellerRecordId to FunctionalGroupName |
| Roy Nabel | 9/6/2012 | Added setPayments and added FunctionalGroupName to several set functions |
| Roy Nabel | 9/12/2012 | Added FunctionalGroupMaps |
| Roy Nabel | 10/5/2012 | Added operation code to all the set methods |
| Roy Nabel | 10/8/2012 | Added R for Operation parameter in several set calls (eg. setTimecards, setUsers, setInvoices) |
| Roy Nabel | 10/18/2012 | Added MobileRecordId to all the major set calls. |
| Roy Nabel | 10/25/2012 | Added getStoreShelfLocations |
| Roy Nabel | 11/2/2012 | Added copyUserRecord for creating customer functional groups |
| Roy Nabel | 11/3/2012 | Added arguments to many set calls |
| Don Zhang | 11/4/2012 | added & modified several alert calls |
| Roy Nabel | 11/6/2012 | Modified argument list for several set calls; removed replace calls in qb service |
| Roy Nabel | 11/9/2012 | Changed setTimecards argument order worker id |
| Roy Nabel | 11/11/2012 | Added orgName, orgNumber to setTerms |
| Don Zhang | 11/25/2012 | Added createSecurityUser, deleteSecurityUser, moveSecurityUser and updateSecurityUser |
| Roy Nabel | 12/4/2012 | In timecard detail replaced TimecardHeaderBarcode with MasterBarcode |
| Roy Nabel | 12/20/2012 | Added createOrganization |
| Roy Nabel | 1/10/2013 | Added createAlertWebFeedback; added Overtime to Timecard Detail CSV in setTimecards |
| Roy Nabel | 1/20/2013 | Added setAlerts, Modified createOrganization adding settings for functional group maps |
| Roy Nabel | 1/22/2013 | Added getAlerts; getStoreItemsQuantityOnHand; modified setAlerts by adding 3 new fields at end of csv |
| Roy Nabel | 1/24/2013 | Added getRecordContentEncryptionMode |
| Roy Nabel | 1/25/2013 | Removed RecordContentEncryptionMode & getStoreItemsQuantityOnHand because other functions did the job. |
| Roy Nabel | 1/26/2013 | Added setDealers, getDealers |
| Don Zhang | 1/27/2013 | Added Alert SMS Carrier to setAlerts |
| Roy Nabel | 1/31/2013 | Added createLease |
| Roy Nabel | 2/1/2013 | Added getJobs |
| Roy Nabel | 2/4/2013 | Added setJobs |
| Roy Nabel | 2/5/2013 | Modified createFileInZipCode added the check boxes arguments and image file instead of html. Removed getJobs, setJobs; added all the Map service calls. |
| Roy Nabel | 2/11/2013 | Added getJobs back in again and added parameter technicianRecordId; added emailToList |
| Roy Nabel | 2/13/2013 | Added getBills, SetBills |
| Roy Nabel | 2/19/2013 | Added getExpenses, SetExpenses, setJobs |
| Roy Nabel | 2/20/2013 | Added org name, org # to setJobs |
| Roy Nabel | 2/21/2013 | Added function group name to set expenses, and set jobs |
| Roy Nabel | 2/22/2013 | Added function group name and item type to setJobs |
| Roy Nabel | 3/1/2013 | Modified setJobs, setBills changed to Header Job Number, Header Bill Number |
| Roy Nabel | 3/5/2013 | Added coding fields for technician timimg to the header after org name and org # of setJobs |
| Roy Nabel | 3/7/2013 | Added MobileRecordId to setJobs detail; SmsMessageToFunctionalGroup; EmailMesssageToFunctionalGroup |
| Roy Nabel | 3/12/2013 | Added getContracts; getFunctionalGroupAlertInfo |
| Roy Nabel | 3/13/2013 | getStoreNumberForLocationRecordId |
| Roy Nabel | 3/14/2013 | Added getRecordIdsByIsa; getRecordsUpdatedByIsa |
| Roy Nabel | 3/18/2013 | Modified setJobs, setBills |
| Roy Nabel | 4/9/2013 | getCalendarEvents, setCalendarEvents, setProspects |
| Roy Nabel | 4/16/2013 | Updated setCalendarEvents |
| Roy Nabel | 5/6/2013 | Added fields to setJobs |
| Roy Nabel | 5/7/2013 | Added fields to setJobs |
| Roy Nabel | 5/26/2013 | Added getEncryptMode, readWebFile, writeWebFile |
| Roy Nabel | 5/29/2013 | Rename setVinStores to setTaxis & added fields |
| Roy Nabel | 6/1/2013 | Added setParts |
| Roy Nabel | 6/5/2013 | Added setJobPhotos |
| Roy Nabel | 6/7/2013 | Added fields to setParts |
| Roy Nabel | 6/8/2013 | Added fields to setJobPhotos |
| Roy Nabel | 6/24/2013 | Added fields to setParts (VIN & Medallion Id) |
| Roy Nabel | 7/16/2013 | Added setFormProspects user web service; added setStores; renamed setProspects to setUserProspects |
| Roy Nabel | 7/17/2013 | setProspects under sales service; remove setFormProspects |
| Roy Nabel | 7/21/2013 | setParts added Vendor Name to csv |
| Roy Nabel | 7/22/2013 | Added VendorName to createPart |
| Roy Nabel | 7/24/2013 | Added Master Barcode to setInvoices |
| Roy Nabel | 8/13/2013 | Added setServiceRequests |
| Roy Nabel | 8/14/2013 | Added getPendingServiceTickets; setServiceTickets |
| Roy Nabel | 8/20/2013 | Added updateSerialNumbers; moveParts |
| Roy Nabel | 9/2/2013 | Modified setRecordCodingFields so that the first 2 arguments are objectId, objectType |
| Roy Nabel | 9/16/2013 | Added fields to setCalendarEvents & setJobs |
| Roy Nabel | 9/25/2013 | Modify setStores to create unique store number |
| Roy Nabel | 10/4/2013 | Added Quantity Received & Quantity Last Received to setPurchaseOrders |
| Roy Nabel | 10/9/2013 | Added ManufacturerSerialNumber to createPart and receivePartDetail. |
| Roy Nabel | 10/11/2013 | Added MobileRecordId and customer:Job to createShipHeader |
| Roy Nabel | 10/18/2013 | Added Directory Style to setStores |
| Roy Nabel | 10/30/2013 | Modified SetServiceTickets to add 2 coding fields: Bad Parts RecordIds; Replacement Parts RecordIds; Modified how setUsers stores new user under Company Name and then if blank Last Name, First Name |
| Roy Nabel | 10/30/2013 | Modified movePart and moveParts adding ScanCode argument and column |
| Roy Nabel | 11/6/2013 | Added MobileInvoiceNumber to createShipHeader |
| Roy Nabel | 11/12/2013 | Added createPartWithCoding |
| Roy Nabel | 11/14/2013 | Added cost to receivePartDetail; added getPhotos and setPhotos |
| Roy Nabel | 11/24/2013 | Added getReceiving, setReceiving, getShipping, setShipping |
| Roy Nabel | 11/25/2013 | Modified 11/24/2013 calls to getCustomerRMAs, getVendorRMAs, setCustomerRMAs, setVendorRMAs |
| Roy Nabel | 11/26/2013 | Modified setCustomerRMAs, setVendorRMAs, getCustomerRMAs, getVendorRMAs |
| Roy Nabel | 11/27/2013 | Modified setCustomerRMAs, setVendorRMAs |
| Roy Nabel | 12/20/2013 | Add setReceiveParts and setShipParts |
| Roy Nabel | 12/24/2013 | Added DeleteRecords; Added coding field to setReceiveParts for Mobile Purchase Order |
| Roy Nabel | 12/29/2013 | Added getSystemStatus |
| Roy Nabel | 1/25/2014 | Added arguments to setShipParts, setCustomerRMAs; added setShipPartsNoReturnData and setCustomerRMAsNoReturnData;  Added arguments to setReceiveParts |
| Roy Nabel | 2/11/2014 | Added setShipPartsSingle, setReceivePartsSingle;  setServiceTicketsSingle |
| Roy Nabel | 3/31/2014 | Added setEventTypes, getEventTypes |
| Roy Nabel | 4/9/2014 | Added red, green, blue to setEventTypes |
| Roy Nabel | 4/10/2014 | Added RGBColorName to setEventTypes |
| Roy Nabel | 4/18/2014 | Added to setCustomerRMAs: Item Number Received Item, Item Number Shipped, Description Item Shipped, Description Item Received |
| Roy Nabel | 5/26/2014 | Added recodeTrackedItem |
| Roy Nabel | 5/29/2014 | Added getShipTrackedStatus |
| Roy Nabel | 6/23/2014 | Added setErrorMessage |
| Roy Nabel | 6/26/2014 | Added Latitude, Longitude to setUsers |
| Roy Nabel | 6/27/2014 | Added notes to setShipParts and setReceiveParts |
| Roy Nabel | 7/7/2014 | Added getFileFolders, getJobsByDateRange, Modified setShipParts (Item Number Shipped, added Item Number Received) |
| Roy Nabel | 7/16/2014 | Added getShipWaybillsNotProcessed |
| Roy Nabel | 7/18/2014 | Added uploadFolderFile |
| Roy Nabel | 8/11/2014 | Added getFilterCategoryPaths |
| Roy Nabel | 8/14/2014 | Added getLibraryCoding |
| Roy Nabel | 8/16/2014 | Added getMatters, setMatters |
| Roy Nabel | 8/18/2014 | Added getNodeInfoByTreeId |
| Roy Nabel | 9/9/2014 | Added key argument to setRecordContent for server side encryption. |
| Roy Nabel | 9/12/2014 | Added mode argument to setRecordContent |
| Roy Nabel | 9/29/2014 | Added findOrganization |
| Roy Nabel | 10/9/2014 | Added createOrganization |
| Roy Nabel | 10/15/2014 | Added setSyncMessage |
| Roy Nabel | 10/24/2014 | Added getTimecardHeaderIdsByDateRange; getTimecardSubmitters |
| Roy Nabel | 11/3/2014 | Added getUserTimecardsByStatusAndLogin; |
| Roy Nabel | 11/11/2014 | Added getFunctionalGroupRecordsList; setAdjustments |
| Roy Nabel | 11/12/2014 | Added arguments to getFunctionalGroupRecordsList; setAdjustments |
| Roy Nabel | 11/17/2014 | Added getNodesByRecordType |
| Roy Nabel | 11/22/2014 | Added Item Discount and Service Discount to setUsers |
| Roy Nabel | 11/26/2014 | Added getRecordCodingByRecordId, getRecordCodingByMobileRecordId |
| Roy Nabel | 11/27/2014 | Added getNodeInfoByByMobileRecordId |
| Roy Nabel | 12/2/2014 | Modified createOrganization to create the local administrator and use web server ip instead of a login/password combo. |
| Roy Nabel | 12/2/2014 | Add getStoreAlertIds, getStoreAlertData |
| Roy Nabel | 12/6/2014 | Added fromDate, toDate to getUserTimecardsByStatusAndLogin |
| Roy Nabel | 12/7/2014 | Added getLatLonByRecordType |
| Roy Nabel | 12/17/2014 | Added setQuickbooksOnlineLog |
| Roy Nabel | 12/18/2014 | Added lat, lon to setStores |
| Roy Nabel | 1/11/2015 | Modified uploadScanLog and more documentation on setRecordCodingFields |
| Roy Nabel | 1/20/2015 | setUserLogin |
| Roy Nabel | 1/21/2015 | getInvoicesByDateAndSalesRep; getTimecardTotalHours |
| Roy Nabel | 2/2/2015 | Added setNodeNameByRecordId |
| Roy Nabel | 2/3/2015 | change setNodeNameByRecordId to setDriectoryNodeName and added arguments |
| Roy Nabel | 2/5/2015 | Added setTrucks |
| Roy Nabel | 3/5/2015 | Added ExpenseName to setBill hdr+detail; getNodeInfoByObjectIdObjectType |
| Roy Nabel | 3/8/2015 | Added MoveOfflineParts |
| Roy Nabel | 3/10/2015 | Added Duration to setCalendarEvents |
| Roy Nabel | 3/20/2015 | Added several new fields to setJobs and setCalendarEvents |
| Roy Nabel | 4/10/2015 | Added Done Dates to setJobs detail |
| Roy Nabel | 4/28/2015 | Added arguments to setTrucks |
| Roy Nabel | 5/6/2015 | Corrected setCalendarEvents to have 43 args. |
| Roy Nabel | 5/18/2015 | Added findTruck |
| Roy Nabel | 5/26/2015 | Added setSalesRecords, setSalesTypes; |
| Roy Nabel | 5/27/2015 | Added AlertEvent to setCalendarEvents |
| Roy Nabel | 5/28/2015 | Renamed AlertEvent to ReminderTime |
| Roy Nabel | 5/31/2015 | Added more reminder fields to setCalendarEvents |
| Roy Nabel | 6/3/2015 | Added getSalesOrders, setSalesOrders |
| Roy Nabel | 6/8/2015 | Added setSensorData |
| Roy Nabel | 6/13/2015 | Added getSensorData; getCalendarEventsByEventType |
| Roy Nabel | 6/25/2015 | Added setFieldBoxes; setProductBoxes; setShippingBoxes |
| Roy Nabel | 6/28/2015 | Added setTruckLogs; setTruckDvirs |
| Roy Nabel | 7/7/2015 | Modified setTruckDvirs so you can have 2 trailers |
| Roy Nabel | 7/13/2015 | Added setPacking |
| Roy Nabel | 7/14/2015 | Added getPacking |
| Roy Nabel | 7/22/2015 | Added arguments to setPurchaseOrders |
| Roy Nabel | 7/28/2015 | Added appendRecordContent |
| Roy Nabel | 8/5/2015 | Added setTruckAlerts, getTruckAlerts |
| Roy Nabel | 8/10/2015 | Added changeItemNumbers |
| Roy Nabel | 8/11/2015 | Added setChangeItemNumbers |
| Roy Nabel | 8/13/2015 | Added getTruckPacking |
| Roy Nabel | 8/14/2015 | Added setProducePacks |
| Roy Nabel | 8/15/2015 | Added getEngineData; getTrailerData |
| Roy Nabel | 8/16/2015 | Added getTruckEngineDetails; getTruckTemperatureDetails |
| Roy Nabel | 8/16/2015 | Updated setProducePacks |
| Roy Nabel | 8/19/2015 | Added getCsvContent |
| Roy Nabel | 8/25/2015 | Added Driver RecordId to setTruckLogs |
| Roy Nabel | 8/26/2015 | Added setTruckEngine, setTruckSensors, getTruckEngineNow, getTruckSensorsNow, getGPSSummaryNow, getGPSDetailNow, createParts |
| Roy Nabel | 8/27/2015 | Added coding field to setTruckLogs |
| Roy Nabel | 9/1/2015 | Added rule and rule driving date coding fields to setTruckLogs |
| Roy Nabel | 9/2/2015 | Modified setTruckLogs removing and adding coding fields |
| Roy Nabel | 9/5/2015 | Added geofence email and sms arguments to setTruckGeofence |
| Roy Nabel | 9/10/2015 | Add arguments to setTrucks |
| Roy Nabel | 9/21/2015 | getTruckGPSNow new call to replace summary and detail |
| Roy Nabel | 9/22/2015 | Added getInventoryCounts |
| Roy Nabel | 9/23/2015 | Added getTrucksInfo |
| Roy Nabel | 9/29/2015 | Added setPriceCodes, setPriceLevels |
| Roy Nabel | 10/6/2015 | Added getTruckLocations, getTruckInfo |
| Roy Nabel | 10/7/2015 | Added 2 arguments to setStores for cold storage |
| Roy Nabel | 10/12/2015 | Added notes to setRecordRequests |
| Roy Nabel | 10/13/2015 | Modified setBoxes to match OCWD |
| Roy Nabel | 10/14/2015 | Added price codes to setSalesOrders and setInvoices, added getTrucksInGeofences; added Date to setPriceCodes; added fields to the setTruckDvirs detail |
| Roy Nabel | 10/16/2015 | Modified setPriceCodes and setPriceLevels |
| Roy Nabel | 10/17/2015 | Modified setPriceLevels |
| Roy Nabel | 10/18/2015 | Added Trip Name to setTruckLogs |
| Roy Nabel | 10/21/2015 | Added Price Level Code & Price Level Description to setPriceCodes |
| Roy Nabel | 10/22/2015 | Added fields to setCentralFiles |
| Roy Nabel | 10/26/2015 | Added setShips |
| Roy Nabel | 11/4/2015 | Added setShipLogs |
| Roy Nabel | 11/5/2015 | Added getTimeAndDistance |
| Roy Nabel | 11/8/2015 | Modified setTruckTrips |
| Roy Nabel | 11/10/2015 | Added setTruckRoutes |
| Roy Nabel | 11/16/2015 | Added setPickLists |
| Roy Nabel | 11/20/2015 | Added Process Name to setInvoices |
| Roy Nabel | 11/21/2015 | Added arguments to setTimecards |
| Roy Nabel | 11/27/2015 | Added getManifest; setWeights |
| Roy Nabel | 12/02/2015 | Added getCycleCount, setCycleCounts |
| Roy Nabel | 12/07/2015 | Added setTemperatures, getTrackingInfo |
| Roy Nabel | 12/08/2015 | Added setMailPackages; getWeights, getTemperatures |
| Roy Nabel | 12/13/2015 | Added Mailcode to setMailPackages and setUsers |
| Roy Nabel | 12/14/2015 | Added setMailHistory; setMailLocations |
| Roy Nabel | 1/4/2016 | Added RouteHeaderRecordId to setTruckLogs |
| Roy Nabel | 1/5/2016 | Added arguments to setTruckTrips |
| Roy Nabel | 2/1/2016 | Added setHarvestSeed and setMillingSeed |
| Roy Nabel | 2/3/2016 | Added setProductionSeed |
| Roy Nabel | 2/19/2016 | Added fields at end of setTimecards detail |
| Roy Nabel | 2/20/2016 | Added fields setShippingSeedForms |
| Roy Nabel | 2/29/2016 | changed setI9LaborForms to master/detail |
| Roy Nabel | 3/8/2016 | Added setFill |
| Roy Nabel | 3/9/2016 | Added alert arguments to setSensorData |
| Roy Nabel | 3/14/2016 | Added setFieldLogs |
| Roy Nabel | 3/21/2016 | Added setW4Forms; setNoticeToEmployeeForms |
| Roy Nabel | 3/24/2016 | Added setW4Forms deductions 10 |
| Roy Nabel | 3/29/2016 | Added getSensorAlerts |
| Roy Nabel | 4/1/2016 | Added setCompressorSlips |
| Roy Nabel | 4/8/2016 | Added arguments to setSensorData and setSensors |
| Roy Nabel | 4/9/2016 | Added setMotors, setPumps, setValves |
| Roy Nabel | 4/11/2016 | Modified setSensorData arguments |
| Roy Nabel | 4/13/2016 | Added Item Name setMotors, setValves, setPumps |
| Roy Nabel | 4/23/2016 | Added cycle arguments to setTruckLogs |
| Roy Nabel | 4/25/2016 | Added HoursRemaining to setTruckLogs |
| Roy Nabel | 5/4/2016 | Added setTruckDriverViolations |
| Roy Nabel | 5/5/2016 | Added itemtype to setTruckDriverViolations and arguments to end of setTruckRoutes |
| Roy Nabel | 5/9/2016 | Added weight and lot to setTruckLogs header |
| Roy Nabel | 5/16/2016 | Added setTrailers |
| Roy Nabel | 5/24/2016 | Added treads to setTrailers |
| Roy Nabel | 5/25/2016 | Added setCityLocators, getCityLocator |
| Roy Nabel | 6/1/2016 | Added setFieldTasks, setFieldOperations |
| Roy Nabel | 6/2/2016 | Added Engine Hours to setTrucks |
| Roy Nabel | 6/3/2016 | Added setWaybills |
| Roy Nabel | 6/5/2016 | Added setFields |
| Roy Nabel | 6/16/2016 | Added fields to setInvoices |
| Roy Nabel | 6/24/2016 | Added setPlantings |
| Roy Nabel | 6/27/2016 | Added Truck Number to setTruckRoutes |
| Roy Nabel | 6/29/2016 | Added getTruckETA |
| Roy Nabel | 7/06/2016 | Modified setUsers |
| Roy Nabel | 7/13/2016 | Added parameters to setFields for circular fields |
| Roy Nabel | 7/16/2016 | Added setTruckRules |
| Roy Nabel | 7/18/2016 | Added setTrailerLoadings |
| Roy Nabel | 7/19/2016 | Added setForklifts and setPalletJacks; added 2 arguments to setUsers |
| Roy Nabel | 7/21/2016 | Added setHarvestWeight |
| Roy Nabel | 7/29/2016 | Modified setHarvestWeight & setTrailerLoadings |
| Roy Nabel | 7/31/2016 | Add setApplesNorthSliceRoomForms |
| Roy Nabel | 8/1/2016 | Added getHarvestWeightsUsersSummary, getHarvestWeightsUsersDetail, getHarvestWeightsFieldsSummary |
| Roy Nabel | 8/3/2016 | Added location, variety to setHarvestWeight |
| Roy Nabel | 8/10/2016 | Added trailer number, location, start date, start time to setBillOfLadings |
| Roy Nabel | 8/12/2016 | Added setUserShipTos |
| Roy Nabel | 8/16/2016 | Added arguments to setFields, setPlantings, setHarvestWeight |
| Randy Nichols | 8/16/2016 | Added sendSensorData |
| Roy Nabel | 8/16/2016 | Added setContainerTypes, setHarvestAreas, setProductCategories, setProductTypes |
| Roy Nabel | 9/5/2016 | Added getServerName |
| Roy Nabel | 9/8/2016 | Added setBOLTruckroute |
| Don Zhang | 9/16/2016 | Added SendEmail |
| Roy Nabel | 9/23/2016 | Added setHarvestMethods, setHarvestOperations |
| Roy Nabel | 10/27/2016 | Added setRanches and removed org coding fields |
| Roy Nabel | 11/1/2016 | Added setVendorsToGrowers; added address fields to setRanches |
| Roy Nabel | 11/3/2016 | Removed setVendorsToGrowers; added Employee Id to setTimecards and setUsers |
| Roy Nabel | 11/4/2016 | Added notes to setpurchaseorders header |
| Roy Nabel | 11/8/2016 | Added notes to setmaildeliverylocations |
| Roy Nabel | 11/17/2016 | Added beds total, beds remaing |
| Roy Nabel | 12/9/2016 | Added setMovePartsViolations for mobile pickup command. |
| Roy Nabel | 12/20/2016 | Added setRecordCodingByRecordId |

# API Data Structures

There are many web service calls that return a NodeInfoBase.

public class NodeInfoBase  
{  
 // data members  
 protected int treeId;  
 protected int objectId;  
 protected String objectType;  
 protected String name;  
 protected int parentTreeId;  
 protected boolean isEncrypted;  
 protected int encryptMode;  
}

The encryption mode has several values.

0 – no encryption is applied

1 – auto encryption/decryption where user id password is used for key with no prompting

2 – keyed encryption/decryption where system will prompt for a key when you decrypt

5 – Adobe encryption

This is the java data structure returned by getRecordContent().  A Jason equivalent is sent back to the client.

public class FileInfo

{  
  private String fileName = null;  
  private String filePath = null;  
  private byte[] bytes = null;  
  private String fileExtension = null;  
}

# Coding Fields

The following are important coding fields that used by the system.

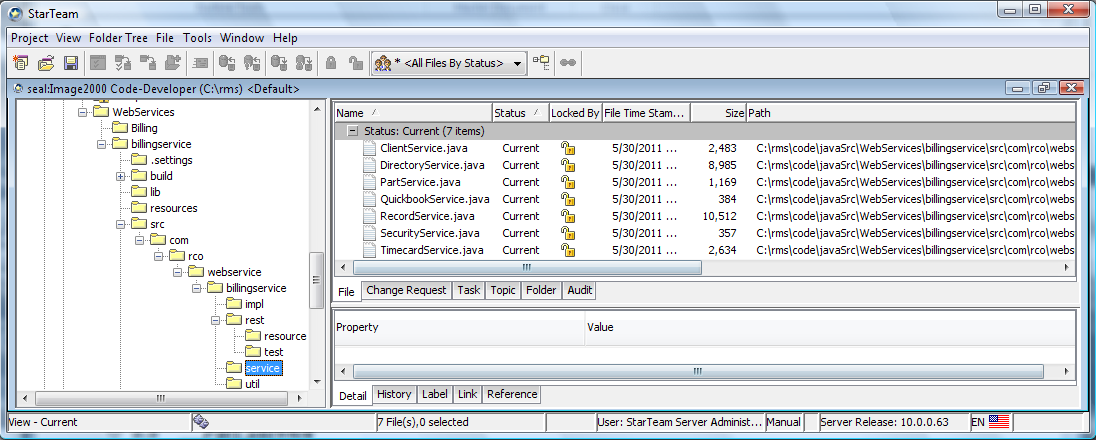
|  |  |
| --- | --- |
| **Coding field** | **Description** |
| RecordId | This is a 64 bit positive number. This is always an 16digit zero padded number on every directory record |
| ItemNumber | This is a 32 bit positive number. This is always an 8 digit zero padded number on every item in the accounting and/or rms system |
| AliasName | This is an alpha-numeric label for the RecordId and must have at least on alphabetic character. This is on a LocationRecordId and is used to make things easier for operator |
| LocationRecordId | This points to a physical directory location object that contains coding fields that describe a location. The system copies all the coding fields to the record except the barcode, recorded, and creation date. |

# Development

This section is for developers so they understand how the code is stored in the Starteam code repository and how to build the project with Eclipse.

## Code Organization in Starteam

The code that implements the json restful web services is checked into starteam (code repository) under the RMS project and then under the folder javaSrc and then under the folder WebServices and then under a folder billingservice. There are several layers of folders under this containing code to implement the service, test the service, resources required to compile the services and utility functions.

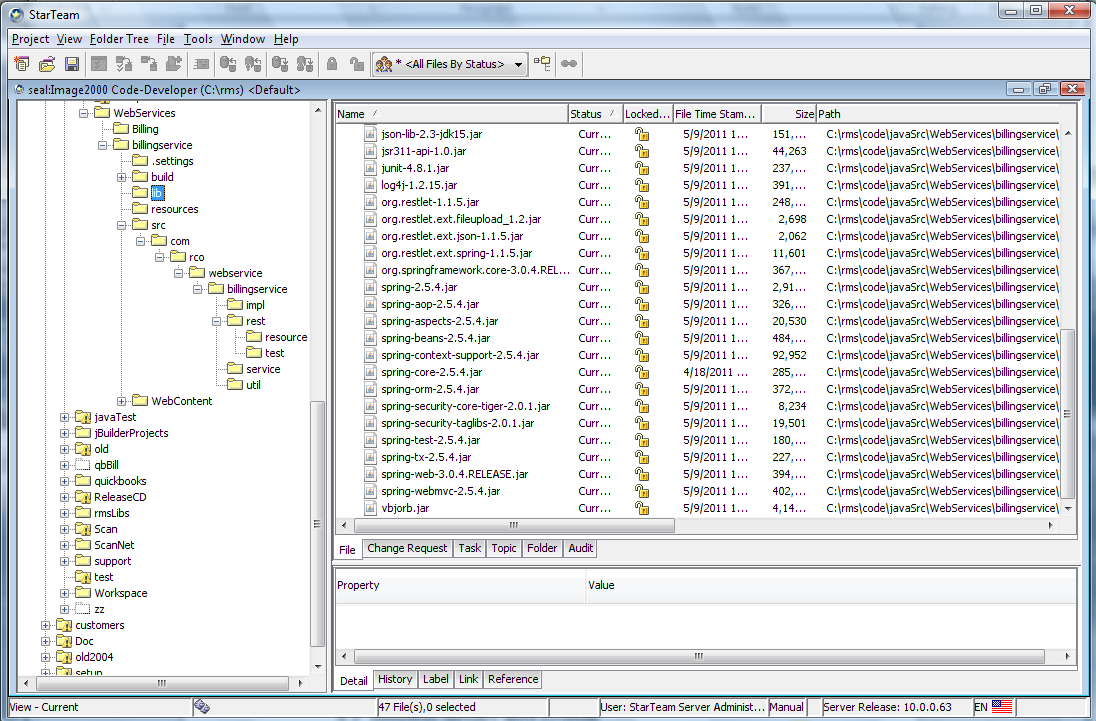


The json restful web services rely heavily on the open source spring framework project.

<http://www.springsource.org/about> (download project from here)

<http://en.wikipedia.org/wiki/Spring_Framework> (Wikipedia discussion)

In addition there are many other opensource libraries used in the project and all of these are stored in the lib folder.

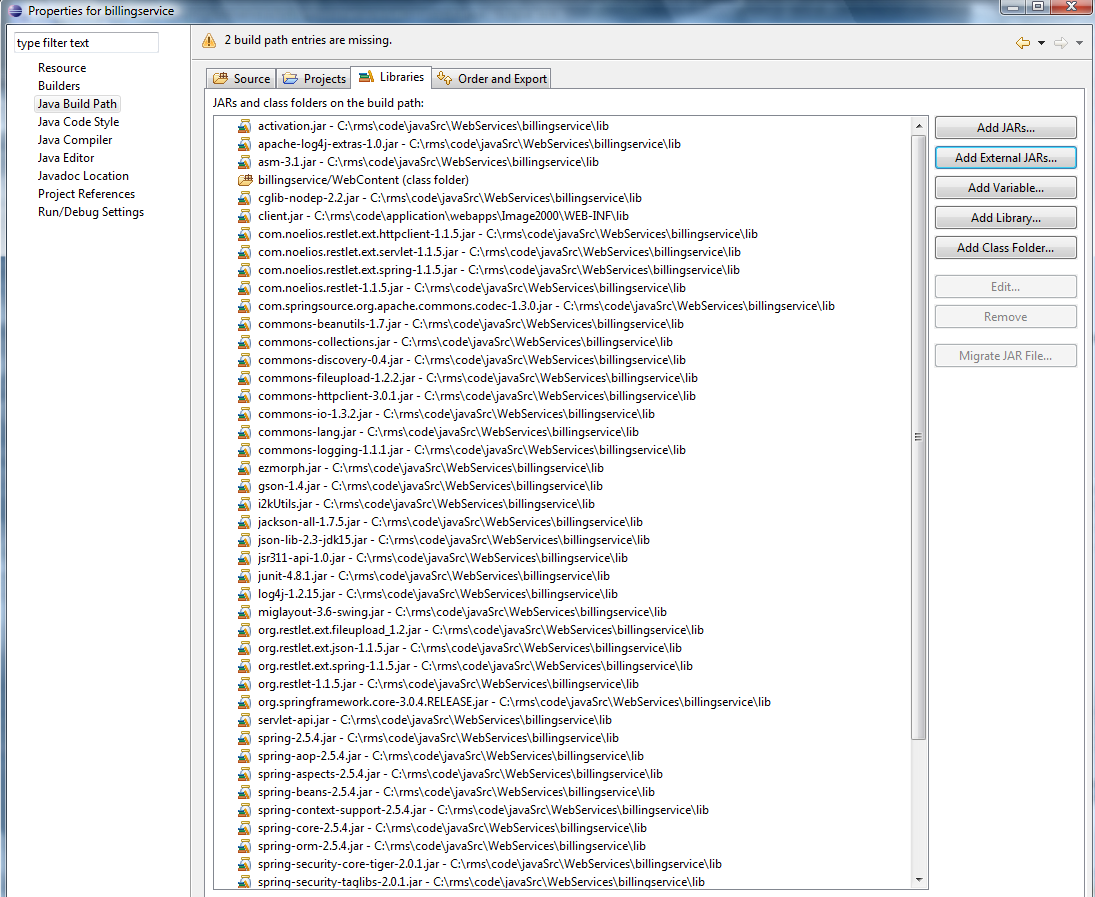


## Development Evironments

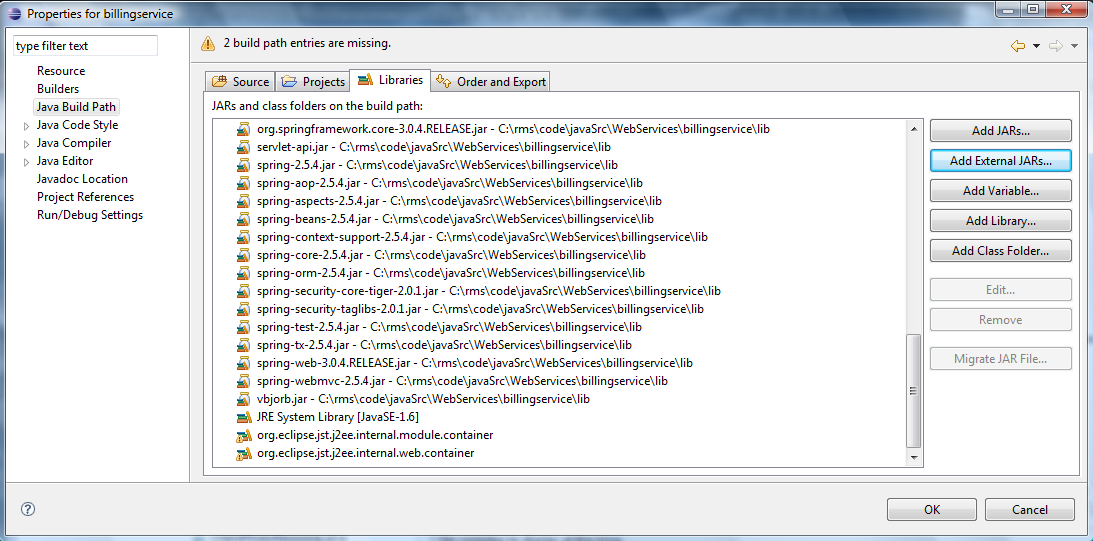
### Restful Web Services Project Development using eClipse

#### Setting up Build Path for Eclipse

You have to setup a billingservice project and the add the external jars to the project. Right click on the billingservice and select properties and the click on build path.

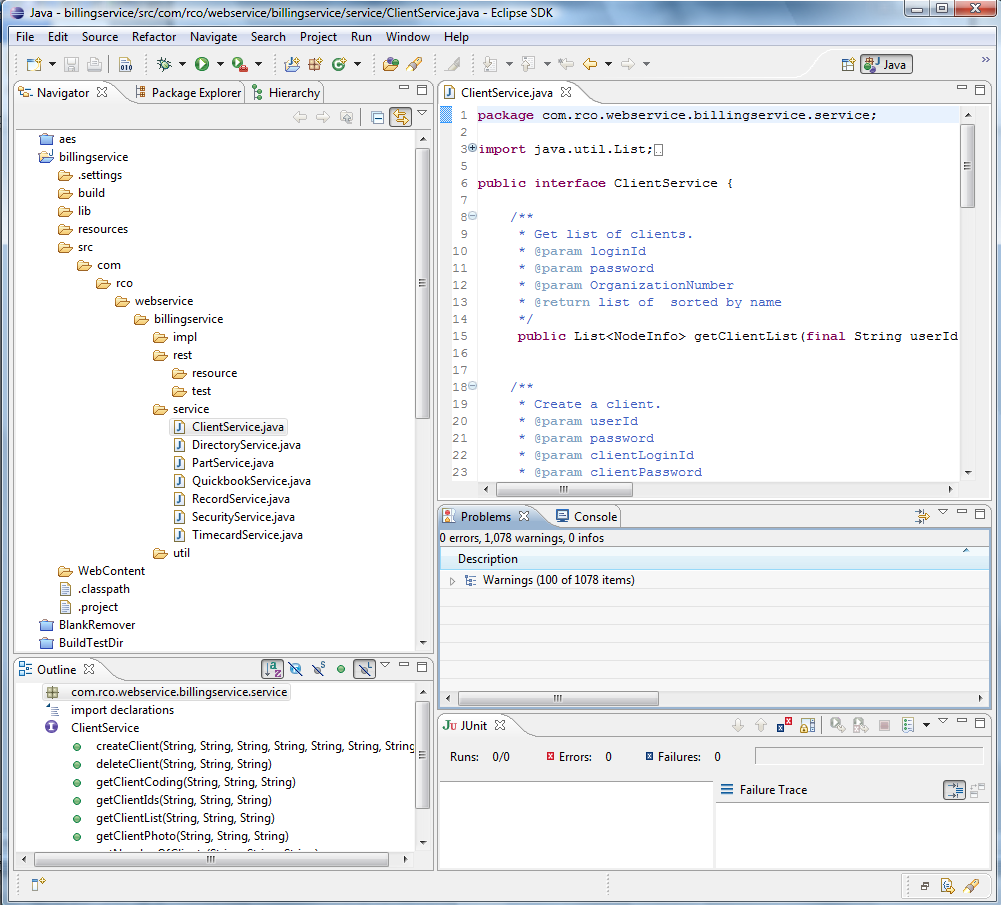


Note that the client.jar is built on seal with the ant build and put in the /rms/code/application/Image2000/webapps/Image2000/web-inf/lib folder.



#### Building project with Eclipse

The project is built in eClipse (on bear there is Europa which is an older version) after you check out the files from starteam.



### Restful Web Services Project Build using Ant.

Software development and source codes modification can be done using any IDEs or editors. After checking source codes into StrTeam, the sources codes can be compiled through the regular build process using Ant build scripts, and packaged ready for deployment.

Under rms/code/antProjects, a sub-directory called restws has been created and checked into StarTeam. The restws directory contains build.xml file, which is an ant build scripts for restful web services. The build.xml defines all the required jar files, which have been checked into rmslibs directory, and automatically put all restful web services java classes into a jar file called restws.jar.

The main build script, build.xml under rms/code/antProjects has been modified to invoke the build script for restful web services, which is basically integrating the restful web service compilation as part of regular ant build process for RMS projects..

## Installing Web Services

When you install the product on a customer machine note the following files in addition to all the other rms files.

Web service config files that are included in the build process are:

~WEB-INF/rest-servlet.xml

~WEB-INF/web.xml

~WEB-INF/classes/billingserviceContent-servoces.xml

~WEB-INF/classes/web-application-content.xml

~WEB-INF/classes/log4j.xml

~WEB-INF/classes/log4j.properties

Outside the build process:

C:/RCOBillingService (including all the subdirectories, and files)

C:/Temp

### Manual Installation

The following steps need to be performed after you build the project in your IDE such as eClipse.

#### Install Java Classes

After compiling billingservice project in your Eclipse environment, copy all java class files with directory path under {your\_workspace}/billingservice/build/classes/com

To

/Image200/webapps/Image200/WEB-INF/classes/com directory in your target server.

#### Install Restful Web Service Configuration Files

1. rest-servlet.xml and web.xml

Those two files, which are checked into StarTeam under application/webapps/Image2000/WEB-INF directory, have been integrated with ant build. Build on seal and update on webserver should automatically deploy the two files into Image2000 environment.

B). Other configuration files

All other configuration files under javaSrc/WebServices/billingservice/resource in StarTeam or under resource directory in your Eclipse need to be copied to

/Image200/webapps/Image200/WEB-INF/classes directory on webserver.

#### Install External Jar files

Copy all the jars in StarTeam under JavaSrc/WebServices/billingservice/lib to

/Image2000/webapps/Image2000/WEB-INF/lib directory.

### Automatic Installation

The build.bat under rms/code/application/Server in StarTeam has been modified to automatically deploy all the restful configuration files and restws.jar file. The restful web services related configuration files are as following:

1. Web.xml

The existing web.xml file has been modified to handle restlet container.

1. Rest\_servlet.xml

This file defines all the restful interfaces

1. Web-application-context.xml and billingserviceContext-services.xml

Those files define the restful web services content and implementation classes.

1. Log4j.xml

This file defines all restful web services logging properties.

## Logging

The web services use the java open source log4j library.

### How to turn on logging

There is a log4j.xml that sets important logging properties. This file is located in the following path: X:\Image2000\webapps\Image2000\WEB-INF\classes

The RESTFUL web service logs are turned on automatically when tomcat server is started.

### Where the logs get stored

There is a log4j.xml that sets important logging properties. This file is located in the following path.

X:\Image2000\webapps\Image2000\WEB-INF\classes

There are 2 file parameters that determine where the current RESTFUL web services logging files are stored.

<param name="File" value="/RCOBillingService/logs/billingservice/billingservice\_info.log"/>

<param name="File" value="/RCOBillingService/logs/billingservice/billingservice\_debug.log"/>

The log files are archived automatically on a daily basis.

## Test Environment

The web services test environment is a windows pc with an internet connection with Cygwin installed and the vi text editor to maintain various test batch scripts.

### Junit Test Cases

All the JUnit test cases developed have been checked into StarTeam as part of the source codes, and each Java class under test package can be run as a JUnit test program in eClipse.

### Batch testing

#### Cygwin

Linux-like environment for Windows making it possible to port software running on POSIX systems (such as Linux, BSD, and Unix systems) to Windows.  
www.**cygwin**.com

#### Batch Scripts

Batch scripts utilizing curl command can be used to test each restful interface and can be scheduled as a job to automatically test one or a group of restful web services. An example of a bach script for testing setRecordContent restful API is following:

curl -k -X POST -F media=@myfile.jpg {webserver}/recordservice/setRecordContent/login/password//objectId/objectType.

# Logistics Web Services

## How It Works

Package status checking process for all the vendors including UPS, FexEx, USPS and DHL consists of a RESTFUL web service, a RESTFUL web service client program and a scheduled job/task running in a Windows environment. A RESTFUL web service has been developed to check package delivery status at real time by calling each vendor web service. The RESTFUL client or consumer program is a Shell script that runs once every hour or however often we would like to schedule it in a windows environment. The whole process starts by a scheduled job/task.

## Carriers

Tracking web services either SOAP or http based XML web service provided by each vendor is one of the business solutions designed for consumers to track delivery status of a shipment or package at real time. The following table shows the different cariers and where you can obtain web tracking information.

|  |  |
| --- | --- |
| **Carrier** | **Description** |
| UPS | Information about UPS tracking program is available at the following link: <https://www.ups.com/upsdeveloperkit> |
| Fedex | Information about UPS tracking program is available at the following link: <https://www.ups.com/upsdeveloperkit> |
| USPS | Information about UPS tracking program is available at the following link: <https://www.ups.com/upsdeveloperkit> |
| DHL | Information about UPS tracking program is available at the following link: <https://www.ups.com/upsdeveloperkit> |

## Restful Web Service

The RESTFUL web service is the key component of the process that invokes UPS SOAP/XML based web service to check the status of a UPS package or shipment based on UPS tracking number(s), move any parts associated with a tracking number/store and update receive date, time and tracking status in RCO System upon detection of a UPS package being delivered.

The RESTFUL web service is available at the following URL:

{webserver}/shipservice/callTrackingServices/{adminLoginId}/{adminPassword}

## Web Service Consumer / Client Program

The RESTFUL Web Service Consumer/Client program is a Shell script developed to run in Cygwin environment. The script utilizes the curl program installed as part of Cygwin to invoke a RESTFUL web service.

The shell script has been deployed on the test server (webserver) at:

C:/cygwin/home/Administrator/jobs/runJob.sh

and content of the script is shown as following:

#!/bin/sh

/cygdrive/c/cygwin/home/Administrator/.bashrc

TODAY=`/usr/bin/date '+%Y%m%d%H%M%S'`

LOG=/home/Administrator/jobs/log/runJob\_$TODAY.log

echo "

---- Log File for runJob.sh ----

started at :

`/usr/bin/date`

" > $LOG

/usr/bin/curl -k -X GET {webserver}/shipservice/callTrackingServices/login/zadmin

echo "

---- Log File for runJob.sh ----

finshed at :

`/usr/bin/date`

" >> $LOG

## Scheduled Job / Task

The runJob.sh is scheduled to run once every hour on webserver through Windows task scheduler.

During each run, a log file is created and stored at C:/cygwin/home/Administrator/jobs/log directory.

## Installation Procedure

Cygwin is a collection of tools which provides a Linux/Unix look and feel environment for Windows

The cygwin installation is as easy as simply clicking on the setup.exe program downloaded from cygwin website. The installation process provides options to pick optional programs/tools to be installed. Please make sure curl program is selected as it is used by the RESTFUL client program or script.

Additional information about cygwin or cygwin installation is available at:

http://www.cygwin.com/

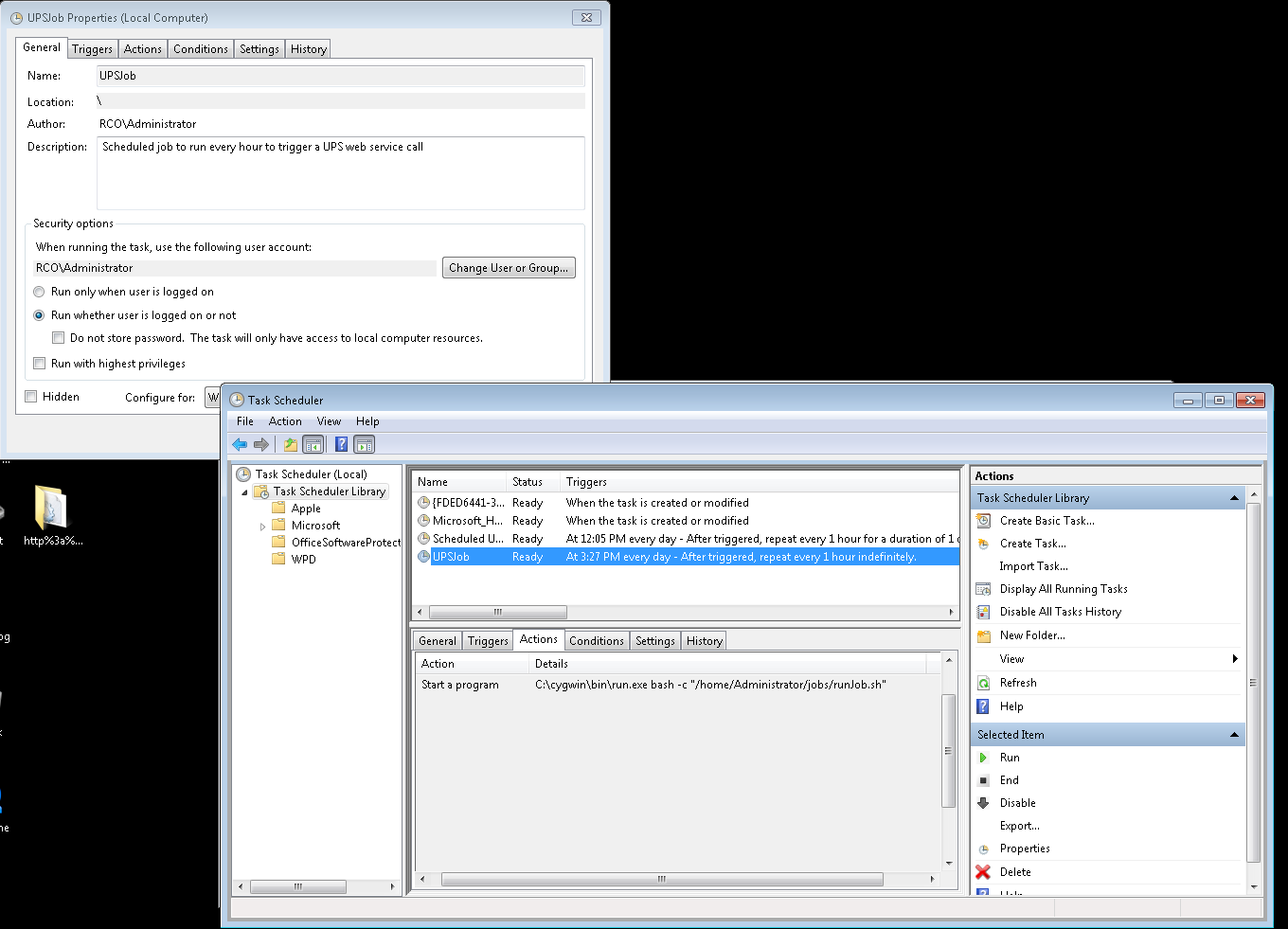
## Restful Web Service Installation

The InvokeUPSServiceJob web service is installed as any other RCO RESTFUL web services which are described more in detail in the RMS Web Services sections.

## Scheduled Task / Job Configuration

The runJob.sh script is currently scheduled to run every hour through Windows Scheduled task Program.

The Scheduled task called UPSJob is configured on webserver as shown in the following screen shot:



Information about how to schedule a task/job in a Windows environment is available at the following link:

http://www.iopus.com/guides/winscheduler.htm

# RMS Web Services

There are several restful json web services that are used to facilitate communication between the various system modules (see Figure 1.0 System Block Diagram).

All queries return text/json and a status code with possible custom response headers. Only post and get queries are supported.

For example, url=”{webserver}/clientservice/{action}/{login}/{password}//{organizationNumber}" Where action = getClientList, createClient, deleteClient, setClientCoding, setClientPhoto, or getClientCoding, etc

## Alert Service

There are many different types of alerts and the system will create records in the directory normally broken out by year-month-day

### createAlertWebFeedback

This call creates a data node in the directory from the web site feedback form.

**POST:**

{webserver}/alertservice/createAlertWebFeedback/{firstName}/{lastName}/{company}/{phone}/{email}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| firstName | Visitor’s first name |
| lastName | Vistor’s last name |
| Company | Visitor’s company’s name |
| Phone | Visitor’s telephone number |
| Email | Visitor’s email address |
| Notes | These are the comments web site visitor left |

### getSensorAlerts

The call getSensorAlerts under alertservice returns sensor data that is out of range for the sensor types specified, date and time range. Normally the time range is the previous 30 minutes.  Returns the objecttypes and objectids.

**GET:**

{webserver}/alertservice/geSensorAlerts/{login}/{password}/{alertType}/{fromdatetime}/{todatetime}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | sendSensorAlerts - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| sensorType | Air Temperature, Humidity,…, all |
| fromdatetime | YYYY-MM-DD HH:MM:SS (24 hour format) |
| todatetime | YYYY-MM-DD HH:MM:SS (24 hour format) |

### setAlerts

The mobile devices can set any number of alerts for any type of user. Alerts can be created, modified or deleted. The alert title is used for the server directory name. There is a chron job that runs every minute and looks to see if there are any alerts that match the current date and time. The chron job sends out emails and sms messages for the alert events that match. In the directory the server creates an alert event (electronic record type) that is based on the year-month and day when the alert is created. If there are several people creating alerts on that day they are all under the same day node.

**POST:**

{webserver}/alertservice/setAlerts/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

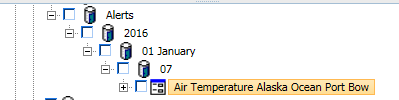
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a header item |
| 3 | objectId | string | If the header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If header exists then this is Invoice Header |
| 5 | Alert Date | String | This is the date the alert will get sent |
| 6 | Alert Time | string | This is the time the alert will get sent |
| 7 | Alert Email Address | string | This is the email address of the person to receive the alert |
| 8 | Alert Email Message | string | This is the email alert message that will get sent |
| 9 | Alert Gmail Login | string | This is the person’s gmail login |
| 10 | Alert Location | string | This is the location for the alert |
| 11 | Alert Notes | Date | This are any alert notes |
| 12 | Alert Notification | string | This is something |
| 13 | Alert Repeat | string | This is the repeat interval if any |
| 14 | Alert SMS Message | string | This is the alert sms message to be sent |
| 15 | Alert SMS PhoneNumber | Date | This is the phone number for the alert sms message |
| 16 | Alert SMS Carrier | String | This is the carrier associated with the phone number |
| 17 | Alert Title | string | This is the title of the alert |
| 18 | Alert URL | string | This is a url for the alert if any |
| 19 | Alert Destination First Name | String | This is the first name of the user for whom the event is associated with. |
| 20 | Alert Destination Last Name | string | This is the last name of the user for whom the event is associated with. |
| 21 | Alert Destination RecordId | string | This is the RecordId of the user for whom the event is associated with. |

The following coding field are put in by the application by looking up the login user coding fields.

|  |  |  |  |
| --- | --- | --- | --- |
|  | AlertCreatorRecordId | string | This is the recordId of the login user who created the alert |
|  | AlertCreatorFirstName | string | This is the first name of the user who created the alert |
|  | AlertCreatorLastName | string | This is the last name of the user who created the alert |

### setSensorAlerts

Alerts can be created, modified or deleted. The directory name is a combination of the SensorType+Area+Location. There is a chron job that runs every minute and looks to see if there are any alerts that match the current date and time. The chron job sends out emails and sms messages for the alert events that match.



**POST:**

{webserver}/alertservice/setSensorAlerts/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a header item |
| 3 | objectId | string | If the header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Organization Name | string |  |
| 8 | Organization Number | string |  |
| 9 | Sensor Name | String |  |
| 10 | SensorRecordId | String |  |
| 11 | SensorType | string |  |
| 12 | Value | Real# |  |
| 13 | Area | string |  |
| 14 | AreaRecordId | string |  |
| 15 | Location | string |  |
| 16 | LocationRecordId | String |  |
| 17 | DateTime | datetime |  |
| 18 | Alert Sent | String |  |
| 19 | Grower | String |  |
| 20 | GrowerRecordId | String |  |
| 21 | Ranch | String |  |
| 22 | RanchRecordId | String |  |
| 23 | Field | String |  |
| 24 | FieldRecordId | String |  |
| 25 | Row | string |  |

### getAlerts

The mobile devices can ask for alerts for a given user and filtered by an alertType and a date range.

**GET:**

{webserver}/alertservice/getAlerts/{login}/{password}/{timestamp}/

Returns:

The objectId, objectType and all the coding fields for each record that matches the criteria.

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| Timestamp | If 0 then return all alerts |

### getStoreLowInventoryItems

This is used to get all the store items where the quantity on hand is less than the min alert level for a given store number.

**GET:**

{webserver}/alertservice/getStoreLowInventoryItems/{login}/{password}/{storeNumber}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| storeNumber | this is the store number where you want to get the alert data |

Returns

storeImageURL, address, city, state, zip, phone

Item Number, Description, Quantity on Hand, Min Quantity on Hand

### getLowInventoryStoreIds

This is used to get all store id’s where at least one item (adjust quantity + quantity on hand) < Min Quantity on Hand that the user has access to.

**GET:**

{webserver}/alertservice/getLowInventoryStoreIds/{login}/{password}/{storeType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | ation code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| storeType | {customer, inventory, vendor} |

Returns

An array of: objectId, objectType, Store Name, StoreNumber, latitude, longitude

## Audit Service

The audit web service creates records in the directory that can latter be used for reporting that sometimes include arithmetic operations. What we want to do in auditing is see who did an operation and when they did it and did it affect the qty on hand of that item. To retrieve itemNumber you can use getPartInfo (../api/PartServices.java)

Please add an audit record for the following function that are location in …/api/PartServices.java. The createMobileScanAudit function is in …/api/CQuickbookServices.java.

movePart

createTrackedPart (location 🡪 new location)

createNonTrackedPart (location 🡪 new location)

The library contains all the accounting item numbers and the coding field Quantity On Hand indicates the count for a given item number. You need to read the item number’s quantity on hand before you execute the operation and then after the operation and update the audit coding fields.

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| old location | This is where the part currently resides |
| new location | Where you are getting the part from that you are going to add to the parent part id. |
| Part or item id | Number of parts to add. A tracked part must have a quantity of 1 |
| Deployment | This is a coding field that help distinguish the status of a part. Is the part in the factory, depot, taxi, vendor |
| scanOperation | This uniquely identifies a client |
| scanDate | This is the current date |
| scanTime | This is the current time |
| QtyScanned | This is the quantity scanned by mobile scan unit |
| QtyOnHandPriorToScan | This is the quantity on hand in library prior to scan |
| QtyOnHandAfterScan | This is the quantity on hand in library after the scan |

### createMobileScanAudit

This creates the mobile scan audit record with the data obtained from uploadScanFile. This will record all the scan operations. Note the description field can be long since you can have

**POST:**

{webserver}/auditservice/createMobileScanAudit/{login}/{password}/{scan operation}/{scan date}/{item number}/{item name}/{QtyScanned}/{QtyOnHandPriorToscan}/{QtyOnHandAfterScan}/{description}

**EXAMPLE:**

{webserver}/auditservice/createMobileScanAudit/login/password//RCVPRT/2011-10-19 09:23:34/960-107/display/2/3/5/randytest/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| scanOperation | This uniquely identifies a client |
| scanDate | This uniquely identifies a client |
| QtyScanned | This is the quantity scanned by mobile scan unit |
| QtyOnHandPriorToScan | This is the quantity on hand prior to scan |
| QtyOnHandAfterScan | This is the quantity on hand prior to scan |
| Description | This contains all scan steps |

### getMobileScanAuditIds

This is used to read the scan audit ids for a given date range for a particular user.

**GET:**

{webserver}/timecardservice/getMobileScanAuditIds/{login}/{password}/{from date}/{to date}

**EXAMPLE:**

{webserver}/timecardservice/getMobileScanAuditIds/login/password//2011-05-05 00:00:00/2011-05-05 00:00:00/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| fromDate | This is the starting rma date you want to include in the query |
| endDate | this is the last rma data you want to include in the query |

## Billing Service

The billing service deals with the management of service jobs

### createOrganization

This call creates a unique organization directory node and sets a unique organization number. The organization name must be unique and will be created unique by the program by adding an integer to the end of the organization name if required.

A local administrator is also created under Organization Name Employees. The program will set the security for the local administrator starting from the root node.

The code will only accept this call from the rco web site with ip = duck2.

Create a storage node with first list of org name if it does not exist from the root node

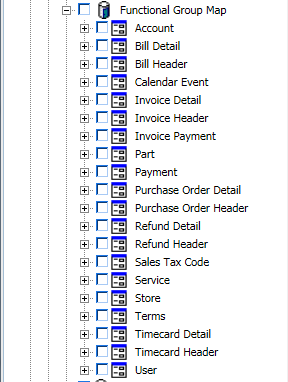
Create a directory node for organization and add a number if needed

Create all the children nodes under the organization

Example

|  |  |  |
| --- | --- | --- |
|  |  |  |

Under the settings there is a storage node called Functional Group Map that is created along with several Functional Group Map record types (see the following figure). The coding fields have to be set for the functional group and user group mappings. Also the Database in the settings has coding fields that have to be set.





**GET:**

{webserver}/billingservice/createOrganization/{login}/{password}{OrganinzationName}/{email}/{password}/{FirstName}/{LastName}/{telephone}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| Login | Admin login |
| Password | Admin password |
| OrganizationName | Name of the new organization to add (must be unique) |
| Email | This will be the login for the local administrator |
| Password | User’s password for authentication |
| FirstName | First name of the local administrator |
| LastName | Last name of the local administrator |
| Telephone | Telephone Number of the local administrator |

### getJobs

This is used to get the open jobs for a given user. A manager can use this call to retrieve the open jobs for a given user they have access to via a functional group.

**GET:**

{webserver}/billingservice/getJobs/{login}/{password}/{technicianRecordId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| technicianRecordId | This can be blank and then it defaults to the login user’s Record Id |

### getJobsByDateRangeAndStatus

This is used to get the all jobs for a given user between a given date range. A manager can use this call to retrieve the open jobs for a given user they have access to via a functional group.

**GET:**

{webserver}/billingservice/getJobsByDateRange/{login}/{password}/{technicianRecordId}/{startDate}/{endDate}/{isOpen}/{Status}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| technicianRecordId | This can be blank and then it defaults to the login user’s Record Id |
| startDate | Starting date you want to get jobs from |
| endDate | Ending date you want to get jobs from |
| isOpen | Field is either yes or no to indicate job is finished or not. |
| Status | This is the job status |

### createFileInZipcode

This creates an eFile lease file in the storage node by zipcode and the directory name is composed of Last Name, First Name – Job Number.

**POST:**

{webserver}/billingservice/createFileInZipcode/{loginId}/{password}/ {jobNumber}/{company}/{firstName}/{lastName}/{address}/{city}/{state}/{zipCode}/{telephone}/{storageName}/{recordType}/{newJobCheckBox}/{repairJobCheckBox}/{agreeCheckBox}/{legalFormNumber}/

File attachment: eFile in html contains check boxes, text, and signature png

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| jobNumber | This is a unique number created by the web portal form WorkOrderJob which is the parent of the WorkOrderDetails |
| Company | This is the company name if any |
| firstName | The person’s first name responsible for signing the lease |
| lastName | The person’s last name responsible for signing the lease |
| Address | The person’s address reponsible for signing the lease |
| City | The person’s city responsible for signing the lease |
| State | The person’s state responsible for signing the lease |
| zipCode | The person’s zip code responsible for signing the lease |
| Telephone | Telephone of the installation or repair |
| storageName | Directory node this is going to be Leases |
| recordType | What eFile record type (example lease) |
| newJobCheckBox | Boolean – true/false |
| repairJobCheckBox | Boolean – true/false |
| agreeCheckBox | Boolean – true/false |
| legalFormNumber | This is the number of the legal form to use. |

### getPhotos

This function retrieves all the objectIds for child photos of a given header

**POST:**

{webserver}/billingservice/getPhotos/{login}/{password}/{objectId}/{objectType}/{headerRecordId}/{isReturnIdInfoOnly}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name = getPhotos |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| objectId | This is the objectId of the header record |
| objectType | This is the objectType of the header record |
| headerRecordId | This is the parent header recordId containing photos |
| isRetunIdInfoOnly | "true" to return array of record id info,  "false" to return array of codingfield data. |

### setPhotos

This function creates/updates job photos under an existing header. The HeaderObjectId and HeaderObjectType must be non-blank arguments. The function will return NodeInfo for the records created. The function getRecordContent is used to set the photo attached to each record.

For the admin interface the directory nodes are named as follows. If the title is blank then the date and time are used.

(header)

Title (or date & time if title blank)

**POST:**

{webserver}/billingservice/setPhotos/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a invoice header item |
| 3 | objectId | string | If the header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Organization Name | string | This is used by the device |
| 8 | Organization Number | string | This is used by the device |
| 9 | HeaderObjectId | string | If the header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 10 | HeaderObjectType | string | If header exists then this is Invoice Header |
| 11 | HeaderBarcode | string | Barcode of the parent |
| 12 | Date | Date | Date photo was taken entered by system |
| 13 | Time | String | Time photo was taken entered by system |
| 14 | Title | String | Title of photo |
| 15 | Notes | String | Notes of photo |
| 16 | Location | String | Location of photo |
| 16 | ItemType | String | This should be photo |

### setJobPhotos

This function creates/updates job photos under an existing job header. The JobHeaderObjectId and JobHeaderObjectType must be non-blank arguments. The function will return NodeInfo for the records created. The function getRecordContent is used to set the photo attached to each record.

For the admin interface the directory nodes are named as follows. If the title is blank then the date and time are used.

(job header)

Title (or date & time if title blank)

**POST:**

{webserver}/billingservice/setJobPhotos/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a invoice header item |
| 3 | objectId | string | If the header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Organization Name | string | This is used by the device |
| 8 | Organization Number | string | This is used by the device |
| 9 | JobHeaderObjectId | string | If the header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 10 | JobHeaderObjectType | string | If header exists then this is Invoice Header |
| 11 | JobHeaderBarcode | string | Barcode of the parent |
| 12 | Date | Date | Date photo was taken entered by system |
| 13 | Time | String | Time photo was taken entered by system |
| 14 | Title | String | Title of photo |
| 15 | Notes | String | Notes of photo |
| 16 | Location | String | Location of photo |

### setJobs

This function creates/updates jobs (header and or details). Jobs are normally created by the web portal job header form and job detail form. The purpose of this call is that you can quickly create test data under the Jobs storage node which is under the organization node. The attached csv file uses a CRLF character to separate records. The csv file must contain at least one header line and zero or more detail lines. Each header line must start with an H character and each detail line must start with the D character and must contain the Invoice Number to link the detail and header record.

The current rule for setJobs(), etc, is that if a CSV row has non-blank ObjectId and ObjectType values, it the corresponding record will be updated.  If those values are blank, a new record will be created in the database.

For the admin interface the directory nodes are named as follows.

Last Name, First Name Job # (job header)

Service Item # Description (job detail)

On desktop applications the JobParentRecordId will be used for linkage whereas the mobile devices will use JobParentMobileRecordId.

**POST:**

{webserver}/billingservice/setJobs/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Account Number | String | This comes from a remote accounting system |
| 8 | Job Number | Auto | This number is assigned by the system |
| 9 | Dispatch Date | Date | Date when dispatcher took the call |
| 10 | Dispatch Time | String | This is when the dispatcher received the job call |
| 11 | First Name | string | This is the job customer first name |
| 12 | Last Name | string | This is the job customer last name |
| 13 | Address1 | string | This is the address of the customer |
| 14 | City | String | This is the city customer is located in |
| 15 | State | string | This is the state customer is located in |
| 16 | ZipCode | string | This is the customer’s zip code |
| 17 | Phone | string | This is the cusomter’s telephone number |
| 18 | Technician RecordId | String | This is the record id of the technician assigned |
| 19 | Technician First Name | string | Assigned technicians first name |
| 20 | Technician Last Name | string | Assigned technician’s last name |
| 21 | Skill Level | string | Level of skill required to do the job |
| 22 | Status | string | Scheduled, Started, Done, Help, Delayed |
| 23 | Latitude | string | This is the gps latitude coordinate of the job |
| 24 | Longitude | string | This is the gps longitude coordinate of the job |
| 25 | Description | string | Short description of the job |
| 26 | Open | string | List box yes=job not done no=job done |
| 27 | Organization Name | string | This is the organization name of the login user |
| 28 | Organization Number | string | This is the organization number of the login user |
| 29 | Start Date Scheduled | Date | Date the techinician started working |
| 30 | Start Time Scheduled | string | Time the techinician started working |
| 31 | Duration Minutes Estimated | Fractional # | This is how long the job should take |
| 32 | Start Date Estimated | Date | This is when the job really gets started |
| 33 | Start Time Estimated | String | This the time when the job really starts |
| 34 | Start Date | Date | This is when the job really gets started |
| 35 | Start Time | String | This the time when the job really starts |
| 36 | End Date | Date | Date when the job actually finished |
| 37 | End Time | String | Time when the job actually finished |
| 38 | Elapsed Minutes | Fractional # | How long the job took in minutes |
| 39 | End Date Estimated | Date | New Estimated date when job will complete |
| 40 | End Time Estimated | String | New Estimated time when job will complete |
| 41 | Cancel Date | Date | Date job got cancelled before completion |
| 42 | Cancel Time | String | Time job got cancelled before completion |
| 43 | Help Needed | Boolean | True = job needs help |
| 44 | CalenderEventRecordId | String | If linked to calendar event |
| 45 | CalenderEventMobileRecordId | String | If linked to calendar event mobile id |
| 46 | DurationInDays | String | How long the jab will last in days |
| 47 | JobRepeatPattern | String | Which days of the week get repeated |
| 48 | JobRepeatSkipDates | String | If you modify one specific event in the pattern |
| 49 | JobRepeatType | String | Never,every day, every week,every two weeks, every month, every year |
| 50 | HelpNeededDates | String | Will contain the dates when the job needed help using a comma to separate multiple dates. |
| 51 | DoneDates | String | Will contain the dates when the job status when was closed using a comma to separate multiple dates. |
| 52 | StartDates | String | Will contain the dates when the job was started using a comma to separate multiple dates. |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” =this is an invoice detail item |
| 3 | objectId | string | This is the objectId for the detail if you are updating a record otherwise it is blank. |
| 4 | objectType | string | This is the objectType for the detail if you are updating a record otherwise it is blank |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | String | This is for security based on login |
| 7 | Account Number | String | This comes from a remote accounting system |
| 8 | Header Job Number | Whole Number | This number is assigned by the system |
| 9 | Serial Number | String | This is a device serial number |
| 10 | Item Number | string | This is the service item number |
| 11 | Item Type | string | This is service or part |
| 12 | Notes | string | Notes related to this job detail |
| 13 | Date | Date | Mobile device auto enters this |
| 14 | Year | String | Strip off year from date |
| 15 | Month | string | Strip off month from date |
| 16 | Day | string | Strip off day from date |
| 17 | Description | string | This is the item # description |
| 18 | Technician RecordId | String | This is the record id of the technician assigned |
| 19 | Technician First Name | string | Assigned technicians first name |
| 20 | Technician Last Name | string | Assigned technician’s last name |
| 21 | Status | string | ??? |
| 22 | Organization Name | string | This is the organization name of the login user |
| 23 | Organization Number | string | This is the organization number of the login user |
| 24 | DoneDates | String | Will contain the dates when the job status when was closed using a comma to separate multiple dates. |

Each data line is separated by a CRLF. Data fields within a line are separated by a comma. All data elements are surrounded by quotes. You can have a header record and no details if you are doing an update on the header. To update a detail you must have the header record.

If the Invoice Numbrer or Mobile Invoice Number exist then you are doing an update operation otherwise you are doing a record creation.

Example of Curl command:

curl -k -X POST -F [media=@fields.txt](mailto:media=@fields.txt) [{webserver}/](https://www.rcofox.com/Image2000/rest/)quickbookservice/setInvoices/login/password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

## Calendar Service

### getCalendarEvents

This is used to get the calendar events for a given user. A manager can use this call to retrieve the open jobs for a given user they have access to via a functional group.

**GET:**

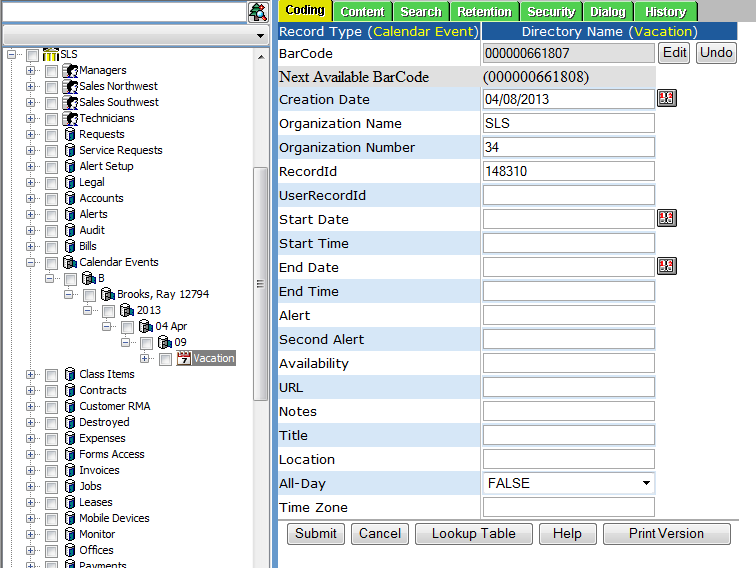
{webserver}/calendarservice/getCalendarEvents/{login}/{password}/{userRecordId}/{startDate}/{endDate}/{eventType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| userRecordId | user’s Record Id |
| startDate | Filter start YYYYMMDD |
| endDate | Filter end YYYYMMDD |
| eventType | Event type |

### setCalendarEvents

Creates one or more calendar events for one or more users in a give organization (login) under a directory node called Calendar Events. There are storage containers for the last letter first letter and then year, month, day the calendar event was created and then the actual calendar event record’s name is form the title coding field.



**POST:**

{webserver}/calendarservice/setCalendarEvents/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a calendar event header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | UserRecordId | String | User who the event is for |
| 8 | Availability | Boolean | Is the service person ready to work or out? |
| 9 | Title | String | Title of the event |
| 10 | Location | String | Location of where the event will occur |
| 11 | All-Day | Boolean | Will event take the whole day |
| 12 | Timezone | string | What timezone will event occur in |
| 13 | Notes | string | Notes about the event |
| 14 | URL | string | Web site address associated with event |
| 15 | Repeat | string | Does the event repeat (daily, weekly, monthly, yearly) |
| 16 | Alert | string | Want an alert sent prior to event |
| 17 | Second Alert | string | Want a second alert sent prior to event |
| 18 | Organization Name | string | This is the organization name of the login user |
| 19 | Organization Number | string | This is the organization number of the login user |
| 20 | Start Date | Date | Date the techinician started working |
| 21 | Start Time | string | Time the techinician started working |
| 22 | End Date | Date | Date the technician finished working |
| 23 | End Time | string | Time the technician started working |
| 24 | Event Type | string | Example job, vacation, medical, personal |
| 25 | CustomerRecordId | string | RecordId of the customer if event type – job |
| 26 | JobParentRecordId | string | RecordId of the job parent record or Timecard Header Record Id |
| 27 | JobParentMobileRecordId | string | Mobile RecordId of the job parent record or Timecard Mobile Record Id |
| 28 | JobDetailRecordId1 | string | RecordId of the job detail record |
| 29 | JobDetailRecordId2 | string | RecordId of the job detail record |
| 30 | JobDetailRecordId3 | string | RecordId of the job detail record |
| 31 | JobDetailRecordId4 | string | RecordId of the job detail record |
| 32 | JobDetailRecordId5 | string | RecordId of the job detail record |
| 33 | Status | string | Status of an event |
| 34 | Mode | string | Timecard, job, normal |
| 35 | RGBColorName | number | Color of the event |
| 36 | EventRepeatType | number |  |
| 37 | EventRepeatPattern | number |  |
| 38 | Duration | string |  |
| 39 | PreviousLinkRecordIds | string |  |
| 41 | MobilePreviousLinkRecordIds | string |  |
| 42 | NextLinkRecordIds | string | This is the recordids of next calendar events |
| 43 | MobileNextLinkRecordIds | string | This is the mobile recordids of next calendar events |
| 44 | EventId | string | Used to link calendar and jobs. |
| 45 | ReminderTime | string | Time before doing the alert in minutes |
| 46 | ReminderEmailSubject | string | This is the email subject line |
| 47 | ReminderEmailMessage | string | This is the email body message |
| 48 | ReminderEmailList | string | This is a series of one or more email addresses separated by a semi colon |

### getEventTypes

This is used to get all the event types for a given organization.

**GET:**

{webserver}/calendarservice/getEventTypes/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| userRecordId | user’s Record Id |
| startDate | Filter start |
| endDate | Filter end |

### setEventTypes

Creates one or more event types for a given organization under a directory node called Event Types.

**POST:**

{webserver}/calendarservice/setEventTypes/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a calendar event header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Organization Name | String | Title of the event |
| 8 | Organization Number | String | Title of the event |
| 9 | Name | String | This is a unique directory node name |
| 10 | Description | String | short description of the event type |
| 12 | Title | Boolean | Event type title like (patient, job, party,…) |
| 13 | Red | String | Color of the event type on calendar |
| 14 | Green | String | Color of the event type on calendar |
| 15 | Blue | String | Color of the event type on calendar |
| 16 | Mode | String | Timecard, job, normal |
| 17 | RGBColorName | String | Name of RGB Color allows custom color |

## Data Service

This is used to extract and manipulate record data from the database to meet various needs.

### getNestedList

**Description:**

Build a nested directory list based on record type. We allow up to 4 directory level branches but you must specific at least one codingFieldName1. This returns an array of strings. For security we use the objectId of the user’s group. If the record (like efile) has a UserGroupId=0 then the record is Public and everybody can see the record. If UserGroupId > 0 then only people with that UserGroupId (this is the internal database ObjectId for the group) can see that record. Remember if you are a super admin or local admin you will not be able to see the record when using the mobile device!

**GET:**

{webserver}/dataservice/getNestedList/{loginId}/{password}/ {objectType } /{recordType}/

{codingFieldName1}/{codingFieldValue1}/

{codingFieldName2}/{codingFieldValue2}/

{codingFieldName3}/{codingFieldValue3}/

{codingFieldName4}/{codingFieldValue4}/

{codingFieldName5}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getDirectoryIds - action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| recordType | This would the recordtype=efile or part |
| codingFieldName1 | This is the top level coding field Record type example codingFieldName1=category. You would leave the codingFieldValue1= (blank) |
| codingFieldValue1 | This value is filled out if you fill out the next codingFieldName2. For example suppose you want to find all subjects where category=legal |
| codingFieldName2 | Subject |
| codingFieldValue2 |  |
| codingFieldName3 | Topic |
| codingFieldValue3 |  |
| codingFieldName4 | Detail |

### getFunctionalGroupSyncRecordTypes

Get the header and detail records for a given functional record id. This call is used by mobile devices to see if there have been any changes to a record type the login user has access to. If the RMS Coding Timestamp is different from what is stored on the mobile’s local database then the detail records are examined and the getRecordIds updated for each record type is called.

**GET:**

{webserver}/dataservice/geFunctionalGroupSyncList/{loginId}/{password}/ {function record id} /

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Functional Record Id | This is the record id of the functional group the user login belongs to. |

### setFunctionalGroupSyncRecordTypes

Set the header and detail records for a given functional record id.

**POST:**

{webserver}/dataservice/geFunctionalGroupSyncList/{loginId}/{password}/ {functional group record id} /

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |
| --- | --- | --- |
| **Argument** | **Data Type** | **Description** |
| H | string | Indicates this is a header item |
| objectId | string | If the header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| objectType | string | If header exists then this is Header |
| Functional Record Id | string | This is the functional group sync record |
| organizationName | string | This is the organization name to whom the customer belongs |
| organizationNum | string | This is the organization number to whom the customer belongs. |

**Detail Line terminated with CRLF**

|  |  |  |
| --- | --- | --- |
| **Argument** | **Data Type** | **Description** |
| D | string | Indicates this is a detail item |
| objectId | string | This is the detail objectId. When you do an update you must have a header. |
| objectType | string | This is the detail objectType |
| Functional Record Id | string | Same as the header Functional Record Id |
| Record Type | string | This is the record type that was touched |
| RMS Modified by User Id | string | This is the login user id |
| RMS Modified on Date | string | Date the record type was touched |
| RMS Modified on Time | string | Time the record type was touched |
| RMS Modified by First Name | string | Login user first name |
| RMS Modified by Last Name | string | Login user’s last name |

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Functional Record Id | This is the record id of the functional group the user login belongs to. |

### getHarvestWeightsFieldsSummary

Get a summary list of weights for a specific field or all fields.

**GET:**

{webserver}/dataservice/getHarvestWeightFieldsSummary/{loginId}/{password}/ {location}/{fromdate}/{todate}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Location | Field(s) |
| fromdate | YYYY-MM-DD |
| todate | YYYY-MM-DD |

### getHarvestWeightsUsersDetail

Get a summary list of weights for a specific user or all users.

**GET:**

{webserver}/dataservice/getHarvestWeightFieldsSummary/{loginId}/{password}/ {UserRecordId}/{location}/{fromdate}/{todate}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| UserRecordId | If blank then get all users |
| Location | Field(s) |
| fromdate | YYYY-MM-DD |
| todate | YYYY-MM-DD |

### getHarvestWeightsUsersSummary

Get a detail list of weights for a specific user.

**GET:**

{webserver}/dataservice/getHarvestWeightUsersSummary/{loginId}/{password}/ {UserRecordId}/{location}/{fromdate}/{todate}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| UserRecordId | Blank means all users |
| Location | Blank means all fields |
| fromdate | YYYY-MM-DD |
| todate | YYYY-MM-DD |

### getTemperatures

Get a list of temperatures based on the input filter arguments.

**GET:**

{webserver}/dataservice/getTemperatures/{loginId}/{password}/ {location}/{fromdate}/{fromtime}/{todate}/{totime}{temperaturedeviceid}/{temperaturedevicename}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Location | General area of temperature device |
| fromdate | YYYY-MM-DD |
| Fromtime | HH:MM:SS |
| todate | YYYY-MM-DD |
| Totime | HH:MM:DD |
| temperaturedeviceid |  |
| temperaturedevicename |  |

### getSensorsUpdatedXFiltered

This call will return sensor data**.**

**GET:**

{webserver}/dataservice/getSensorsUpdatedXFiltered/{login}/{password}/{recordDisplayType}/{maxNumberfullDataRecords}/{maxTimestamp}/{fromDate}/{toDate}/{dateRangeField}/{filterFields}/{strFieldDelim}/{filterValues}/{strValueDelim}/{isFilterByFuncGroups}/{includeFields}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| password | User’s password for authentication |
| recordDisplayType | This is the rms record type name Eg. Timecard Detail |
| maxNumberFullDataRecords | This is the number of records with full data (all coding fields) that the caller can handle. Example suppose you can handle 200 timecards (this could be header and detail records since the system may have been updated) and there are 500 timecards that meet the filter criteria. If more than the fullDataLimit then only 500 timecard header Id’s will be returned. |
| maxTimestamp | This is 0 is to get all records or a positive integer. |
| fromDateTime |  |
| toDateTime |  |
| dateRangeField |  |
| filterFields |  |
| strFieldDelim |  |
| filterValues |  |
| strValueDelim |  |
| isFilterByFuncGroups |  |
| includeFields |  |

### getWeights

Get a list of weights based on the input filter arguments.

**GET:**

{webserver}/dataservice/getWeights/{loginId}/{password}/ {location}/{fromdate}/{todate}/{scaleid}/{scalename}/{processname}/{product}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Location | General area of temperature device |
| fromdate | YYYY-MM-DD |
| todate | YYYY-MM-DD |
| scaleid |  |
| scalename |  |
| processname |  |
| product |  |

### sendSensorData

I’ve implemented a web service API to call from a Particle.io webhook.  The URL is:

<https://www.rcofox.com/Image2000/rest/dataservice/sendSensorData/login/password/particle>

Replace “login/password” with your actual login/password.

The data format for the jsonString in the Particle.publish("MyEvent", jsonString, PRIVATE) call is:

{"coreid":"4c0056001951353338363036","endDateTime":"2016-08-05 00:00:00","intervalSeconds":"60","csvValues":"100., 88., 23.4, .123"}

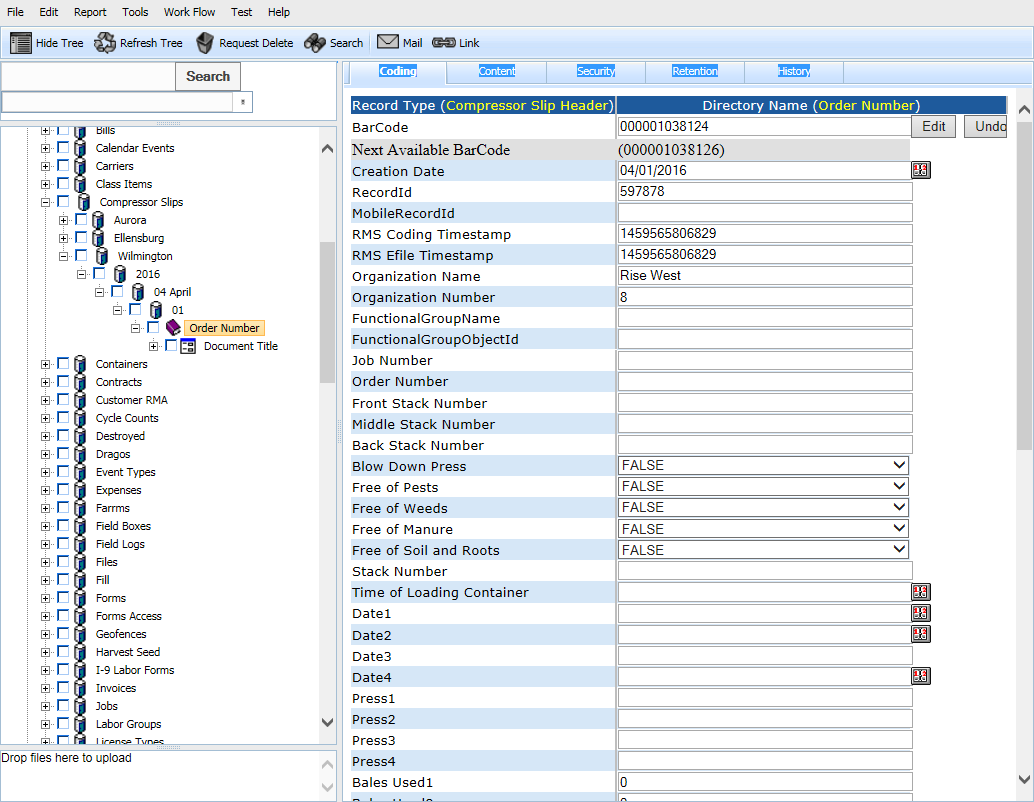
You need to first put the coreid value of your Electron device into the “**Manufacturer Serial Number**” codingfield in the sensor record that is associated with your electron device, under the Sensors node of the Admin Interface Directory Tree.

You should then see records appear under the “Sensor Data” node in the Directory Tree when the event is triggered by your Particle device.

There is a web service test under the Admin Interface: Test | Webservice Test | dataservice | sendSensorData.  You can cut and paste a JSON string into the last parameter (data) to test a JSON string of the above format.  When it asks for a file to upload, just ignore it and press submit.

### setCompressorSlips

This creates/updates one or more compressor slips under organization then Compressor slips first the location then year, month and day followed by the form record with a directory structure like in the following figure.



**POST:**

{webserver}/dataservice/setCompressorSlips/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

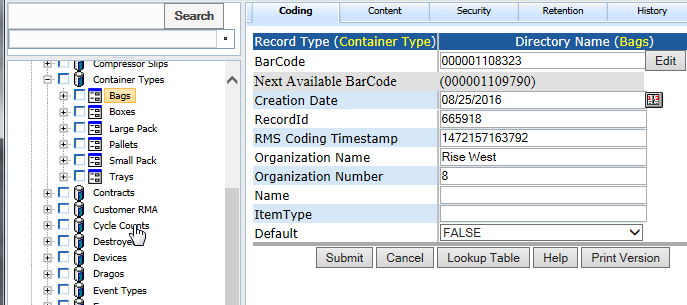
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Job Number | String |  |
| 10 | Order Number | String |  |
| 11 | Front Stack Number | String |  |
| 12 | Middle Stack Number | String |  |
| 13 | Back Stack Number | String |  |
| 14 | Blow Down Press | String |  |
| 15 | Free of Pests | String |  |
| 16 | Free of Weeds | String |  |
| 17 | Free of Manure | String |  |
| 18 | Free of Soil and Roots | String |  |
| 19 | Stack Number | String |  |
| 20 | Time of Loading Container | String |  |
| 21 | Date1 | String |  |
| 22 | Date2 | String |  |
| 23 | Date3 | String |  |
| 24 | Date4 | String |  |
| 25 | Press1 | String |  |
| 26 | Press2 | String |  |
| 27 | Press3 | String |  |
| 28 | Press4 | String |  |
| 29 | Bales Used1 | String |  |
| 30 | Bales Used2 | String |  |
| 31 | Bales Used3 | String |  |
| 32 | Bales Used4 | String |  |
| 33 | Operator1 | String |  |
| 34 | Operator2 | String |  |
| 35 | Operator3 | String |  |
| 36 | Operator4 | String |  |
| 37 | Stack Used1 | String |  |
| 38 | Stack Used2 | String |  |
| 39 | Stack Used3 | String |  |
| 40 | Stack Used4 | String |  |
| 41 | Total Bale Count in Container | String |  |
| 42 | YG1 Number | Date |  |
| 43 | YG2 Number | String |  |
| 44 | Gross Weight | String |  |
| 45 | Tare Weight | String |  |
| 46 | Net Weight | Date |  |
| 47 | Total Short Tons | String |  |
| 48 | Seal Number | String |  |
| 49 | Weight Ticket Number | String |  |
| 50 | Overall Quality | Date |  |
| 51 | Moisture Percentage | String |  |
| 52 | Comments | String |  |
| 53 | Quality Color | String |  |
| 54 | Quality Leaf | Date |  |
| 55 | Quality Grass Sorted | String |  |
| 56 | Quality Burn Sorted | String |  |
| 57 | Quality Texture | String |  |
| 58 | Quality Mold Sorted | Date |  |
| 59 | Quality Dirt Sorted | String |  |
| 60 | Quality Wet Sorted | Date |  |
| 61 | Location | string |  |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D”=this is a invoice detail item |
| 3 | objectId | string | This is the timecard detail objectId. When you do an update you must have a header. |
| 4 | objectType | string | This is the timecard detail objectType |
| 5 | MobileRecordId | string | This is the group responsible for the record |
| 6 | FunctionalGroupName | String | This is used by the local device database and is generated by deviceid+timestamp |
| 7 | organizationName | string | This is the organization name to whom the customer belongs |
| 8 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 9 | Document Title | string |  |
| 10 | Document Type | String |  |
| 11 | Document Date | Date |  |
| 12 | Signature Date | Date |  |
| 13 | Signature Name | String |  |
| 14 | Description | string |  |
| 15 | ItemType | string |  |
| 16 | Reviewed By | String |  |
| 17 | Reviewed Date | Date |  |

### setContainerTypes

This call creates/updates container types as shown in the following figure. The directory name is the name coding field.



**POST:**

{webserver}/dataservice/setContainerTypes/{login}/{password}/

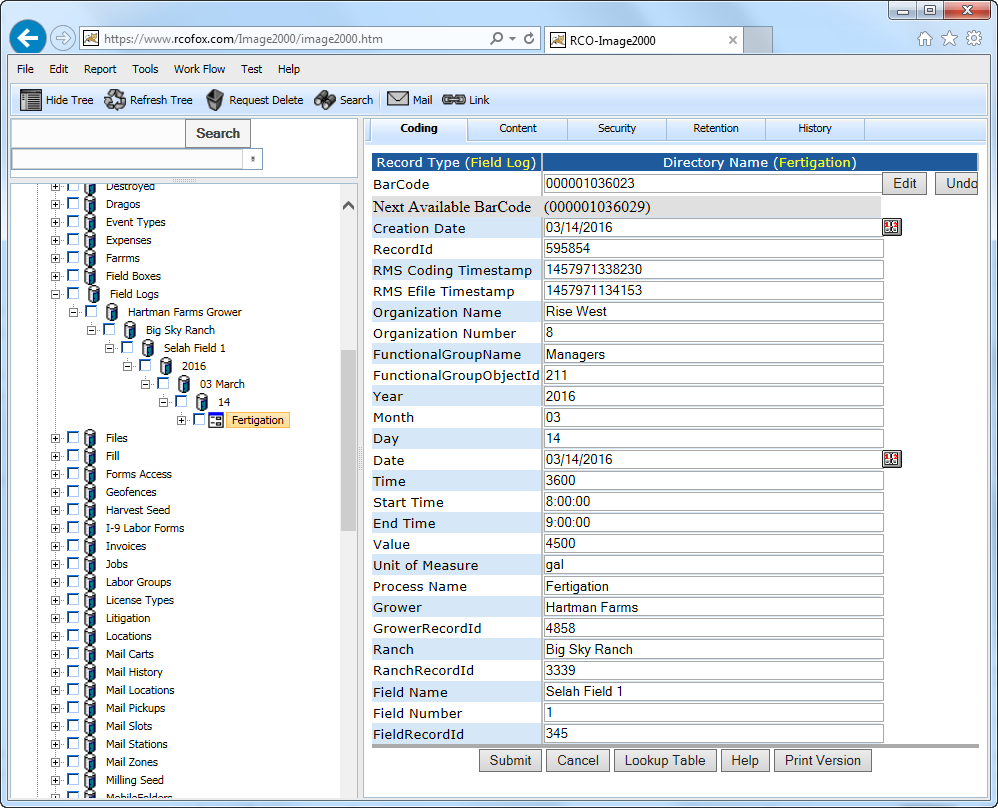
**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Name | String | Type of container {bag, box, bin, truck, pallet} |
| 10 | ItemType | String |  |

### setFieldLogs

This call records data on various processes performed on the field typically at 1 week intervals. The directory structure is grower, ranch, field, year, month, day and process as shown in the following figure.



**POST:**

{webserver}/dataservice/setFiieldLogs/{login}/{password}/

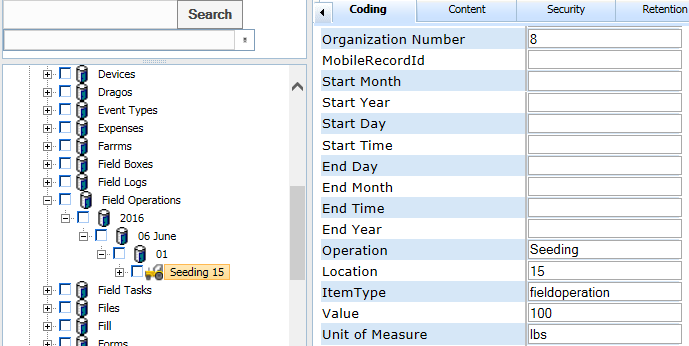
**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Date | Date |  |
| 10 | Time | string |  |
| 11 | Start Time | string |  |
| 12 | End Time | string |  |
| 13 | Value | string |  |
| 14 | Unit of Measure | string |  |
| 15 | Process Name | string |  |
| 16 | Grower | string |  |
| 17 | GrowerRecordId | String |  |
| 18 | Ranch | String |  |
| 19 | RanchRecordId | String |  |
| 20 | Field Name | string |  |
| 21 | Field Number | String |  |
| 22 | FieldRecordId | String |  |

### setFarmOperations

This call creates/updates farm operations as shown in the following figure. The directory name is operation location.



**POST:**

{webserver}/dataservice/setFarmOperations/{login}/{password}/

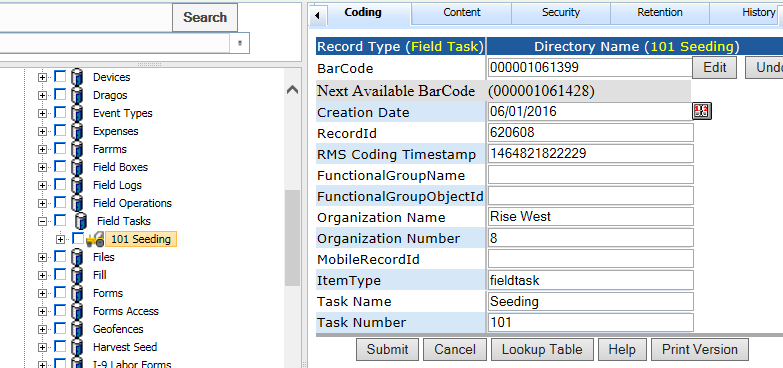
**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Farm Task | Date | This is from the farm tasks list |
| 10 | FarmTaskRecordId | Date | This is from the farm tasks list |
| 11 | Location | string |  |
| 12 | LocationRecordId | string | This will point to a field and the org info fields get copied |
| 13 | ItemType | string |  |
| 14 | Value | string |  |
| 15 | Unit of Measure | string |  |
| 16 | StartDateTime | string |  |
| 17 | EndtDateTime | string |  |
| 18 | Grower | String |  |
| 19 | GrowerRecordId | String |  |
| 20 | Ranch | String |  |
| 21 | RanchRecordId | String |  |
| 22 | Field | String |  |
| 23 | FieldRecordId | String |  |
| 24 | Product Category | String |  |
| 25 | Product Type | String |  |

### setFarmTasks

This call creates/update a lis of farm tasks. The directory structure is a linear list shown in the following figure where you have



**POST:**

{webserver}/dataservice/setFarmTasks/{login}/{password}/

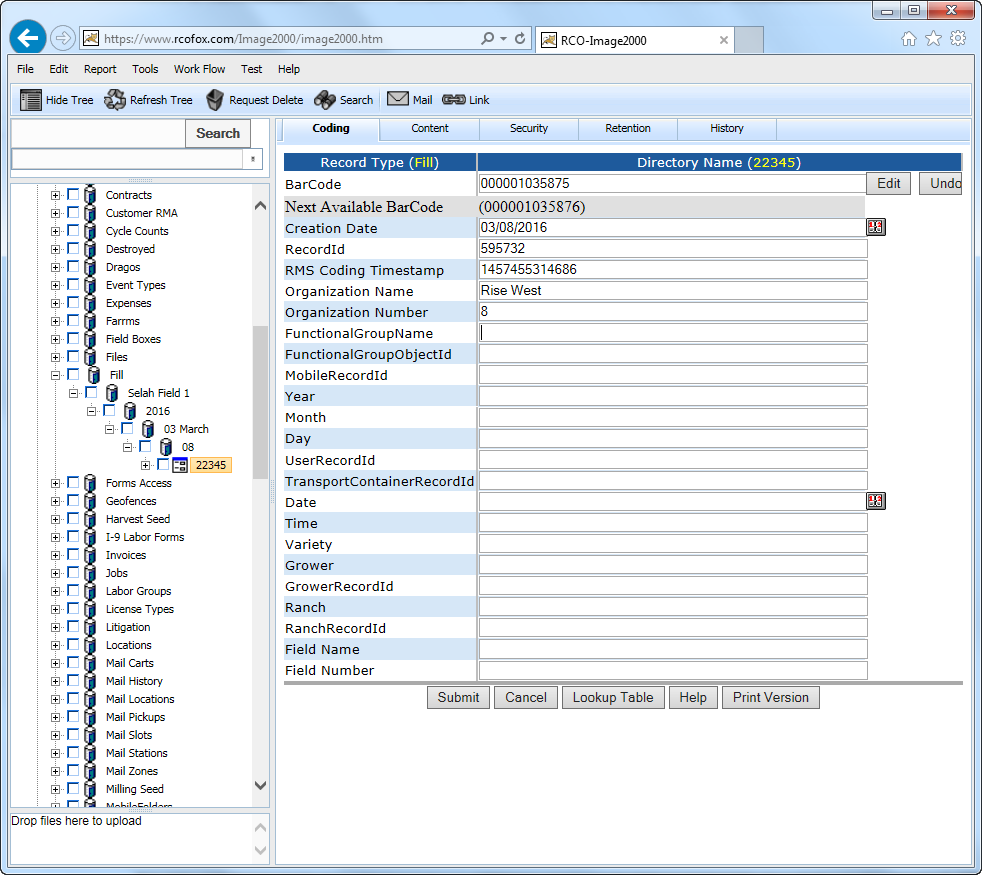
**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | ItemType | Date |  |
| 10 | Task Name | string |  |
| 11 | Task Number | string |  |

### setFill

This call is used to collect the piece quantity of a labor for a given field. Data is stored by the field, year, month, day and transport container record id. The directory name uses the transport container record id.



**POST:**

{webserver}/dataservice/setFill/{login}/{password}/

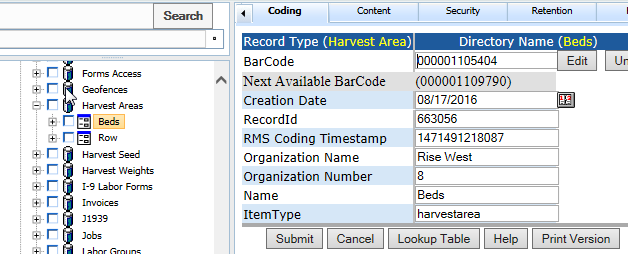
**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | UserRecordId | string |  |
| 10 | TransportContainerRecordId | string |  |
| 11 | Date | string |  |
| 12 | Time | string |  |
| 13 | Variety | string |  |
| 14 | Grower | string |  |
| 15 | GrowerRecordId | String |  |
| 16 | Ranch | String |  |
| 17 | RanchRecordId | String |  |
| 18 | Field Name | string |  |
| 19 | Field Number | String |  |
| 20 | Weight | number |  |

### setHarvestAreas

This call creates/updates harvest areas as shown in the following figure. The directory name is the name coding field.



**POST:**

{webserver}/dataservice/setHarvestAreas/{login}/{password}/

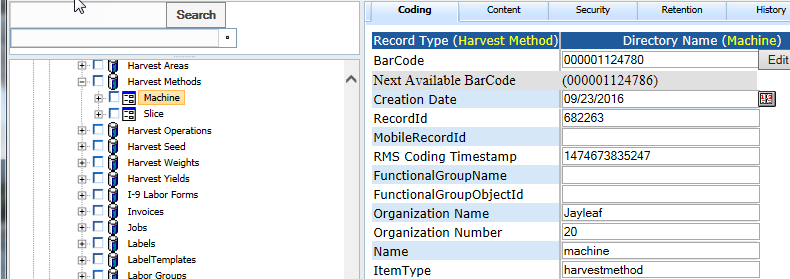
**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Name | String | Type of container {bag, box, bin, truck, pallet} |
| 10 | ItemType | String |  |

### setHarvestMethods

This call creates/updates harvest methods as shown in the following figure. The directory name is the name coding field.



**POST:**

{webserver}/dataservice/setHarvestMethods/{login}/{password}/

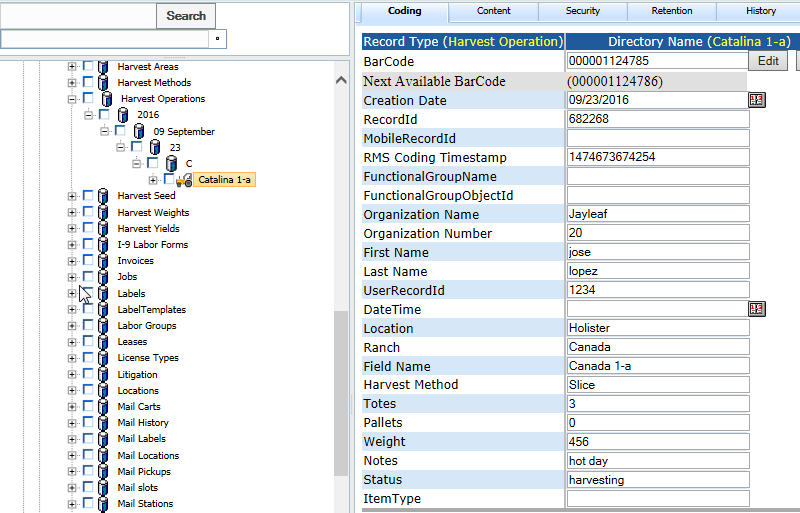
**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Name | String | Type of container {bag, box, bin, truck, pallet} |
| 10 | ItemType | String |  |

### setHarvestOperations

This call creates/updates harvest operations as shown in the following figure. The directory name is the field name coding field. You break things out by year, month and day you did the operations, then the first letter of the field name and then the field name then the data record.



**POST:**

{webserver}/dataservice/setHarvestOperations/{login}/{password}/

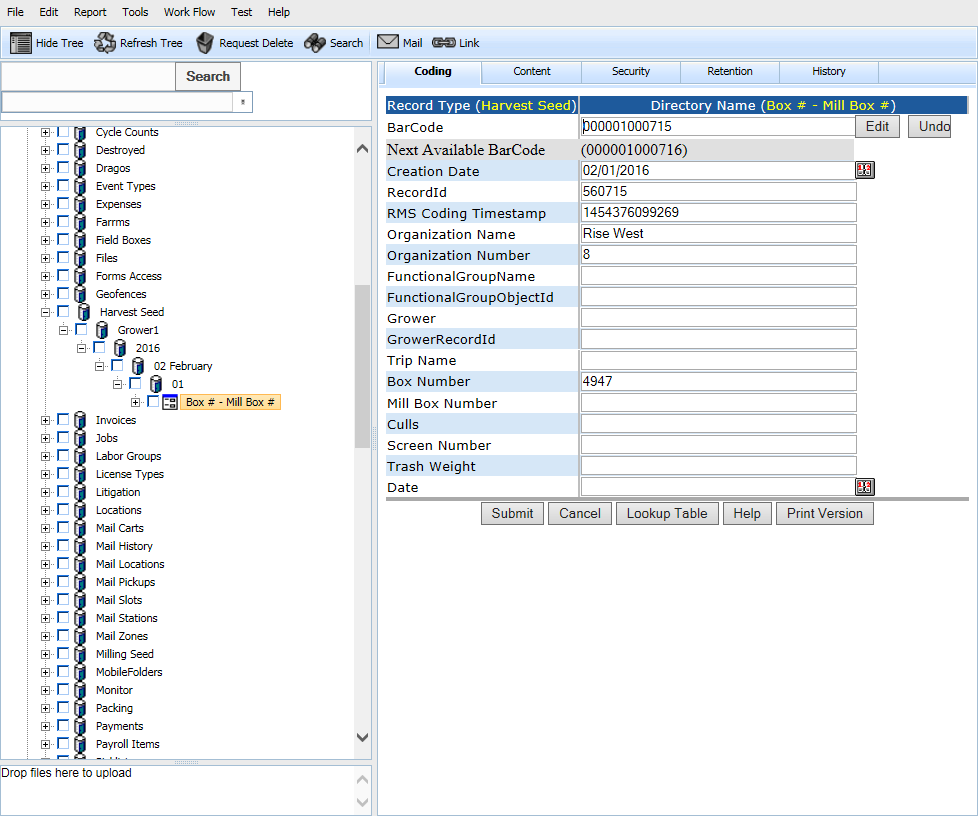
**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | DateTime | String |  |
| 10 | Grower | String |  |
| 11 | GrowerRecordId | String |  |
| 12 | Ranch | String |  |
| 13 | RanchRecordId | String |  |
| 14 | Field Name | String |  |
| 15 | FieldRecordId | String |  |
| 16 | First Name | String |  |
| 17 | Last Name | String |  |
| 18 | UserRecordId | String |  |
| 19 | Location | String |  |
| 20 | Harvest Method | String |  |
| 21 | Totes | String |  |
| 22 | Pallets | String |  |
| 23 | Weight | String |  |
| 24 | Notes | String |  |
| 25 | Status | String |  |
| 26 | ItemType | String |  |
| 27 | Acres | String |  |
| 28 | Beds | String |  |

### setHarvestSeed

This creates a harvest seed record under Harvest Seed then year, month and day with a directory structure like in the following figure.



**POST:**

{webserver}/dataservice/setMillingSeed/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Grower |  |  |
| 10 | GrowerRecordId | String |  |
| 11 | Truck Name | String |  |
| 12 | Box Number | string |  |
| 13 | Mill Box Number | string |  |
| 14 | Mill Weight | string |  |
| 15 | Culls | string |  |
| 16 | Screen Number | String |  |
| 17 | Trash Weight | String |  |

### setHarvestYields

Set Harvest Yields for a given field. The records are stored by Location then field name then year month day and the harvest yield record.

|  |
| --- |
| The mobile or web puts in the following  Harvest Yield |
| Plant SeedPounds |
| Planted Acres |

Then the rms server computes the following

Yield Per Acre – this is the total harvested weight per acre

Yield Per Acre-Seed-Pound

Yield Per Seed-Pound

**POST:**

{webserver}/dataservice/setHarvestYields/{login}/{password}/

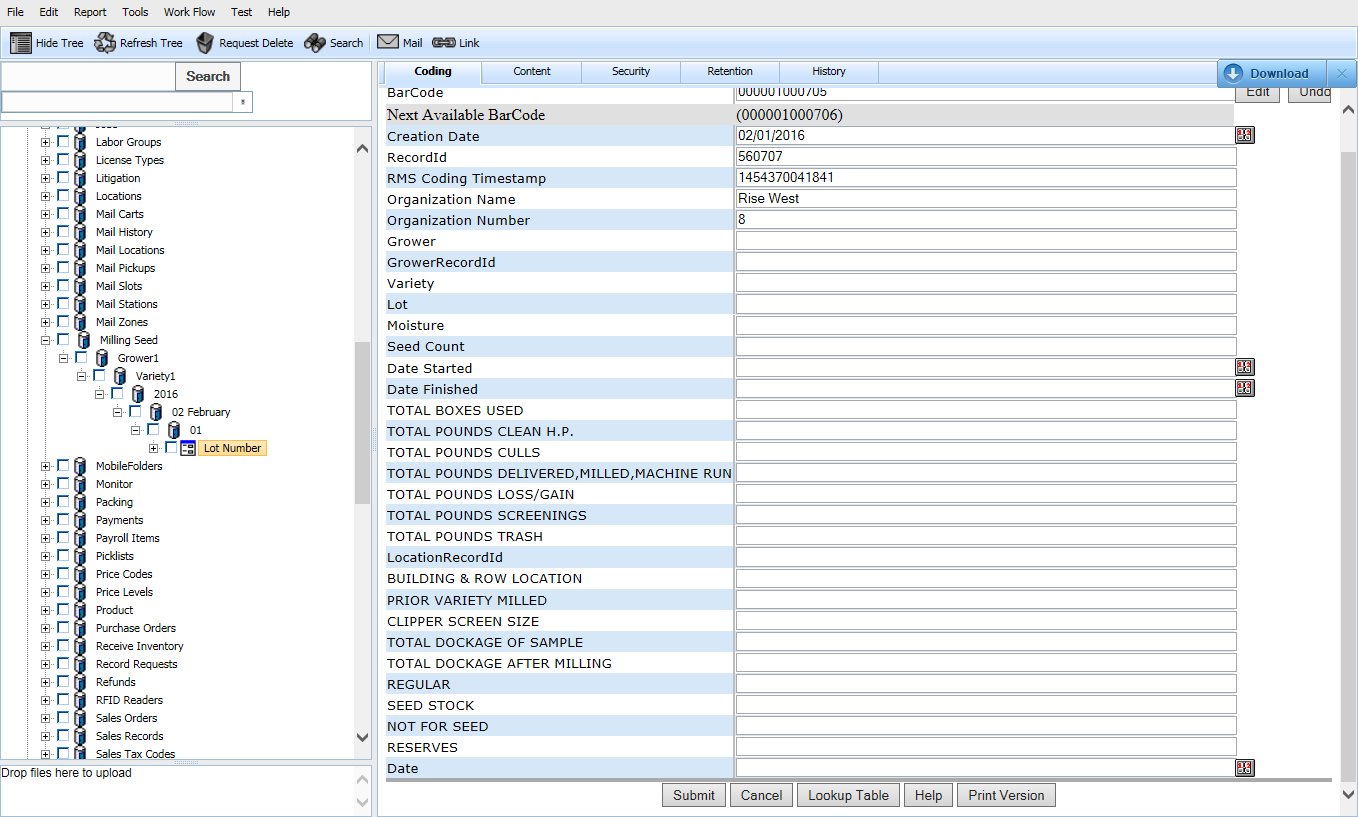
**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | string |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Harvest Date | Date |  |
| 10 | Location | String |  |
| 11 | LocationRecordId | String |  |
| 12 | Field Name | String |  |
| 13 | Field Number | String |  |
| 14 | FieldRecordId | String |  |
| 15 | Product Category | String |  |
| 16 | Product Type | String |  |
| 17 | Harvest Yield | String | This is total pounds harvested |
| 18 | Plant SeedPounds | String | This is the total weight of seeds planted in the field |
| 19 | Planted Acres | String | Total number of acres planted |
| 20 | ItemType | String |  |
| 21 | Status | String |  |

### setMillingSeed

This creates a milling record under Milling Seed then Growerand has the following directory structure.



**POST:**

{webserver}/dataservice/setMillingSeed/{login}/{password}/

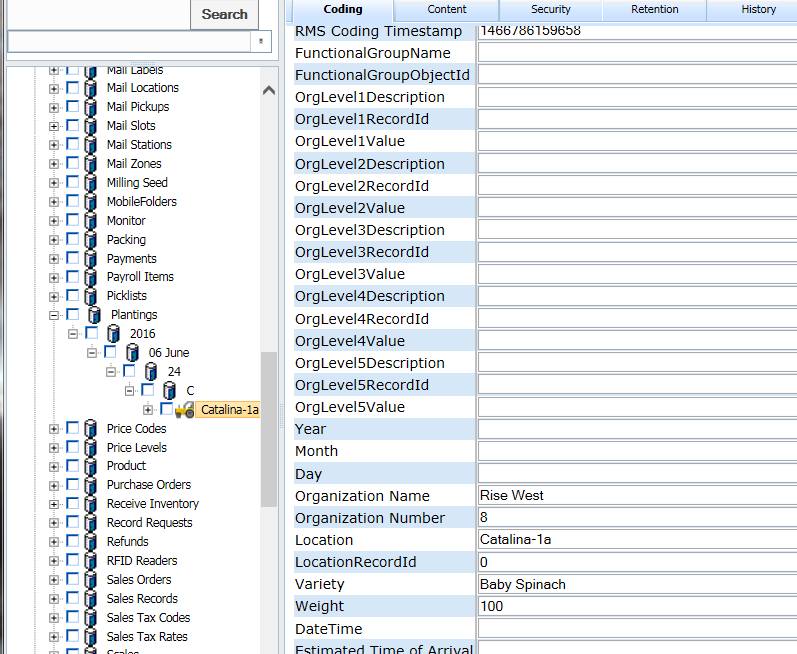
**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Grower |  |  |
| 10 | GrowerRecordId | String |  |
| 11 | Variety | string |  |
| 12 | Lot | string |  |
| 13 | Moisture | string |  |
| 14 | Seed Count | string |  |
| 15 | Date Started | string |  |
| 16 | Date Finished | String |  |
| 17 | Total Boxes Used | String |  |
| 18 | Total Pounds Clean HP | Number |  |
| 19 | Total Pounds Cull | Number |  |
| 20 | Total pounds Delivered | Number |  |
| 21 | Total Pounds Loss or Gain | Number |  |
| 22 | Total Pounds Screenings | Number |  |
| 23 | Total Pounds Trash | Number |  |
| 24 | LocationRecordId | string |  |
| 25 | Building and Row Location | string |  |
| 26 | Prior Variety Milled | String |  |
| 27 | Clipper Screen Size | String |  |
| 28 | Total Dockage Of Sample | String |  |
| 29 | Total Dockage After Milling | string |  |
| 30 | Regular | Number |  |
| 31 | Seed Stock | Number |  |
| 32 | Not For Seed | Number |  |
| 33 | Reserves | Number |  |
| 34 | Date | Date |  |

### setPlantings

This creates/updates a planting record under Plantings then year, month and day, first letter of Location with a directory structure like in the following figure.



**POST:**

{webserver}/dataservice/setPlantings{login}/{password}/

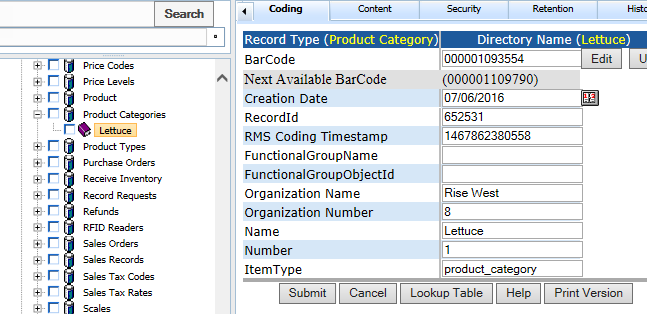
**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | DateTime | string | This is date field was planted |
| 10 | Grower | string |  |
| 11 | GrowerRecordId | String |  |
| 12 | Ranch | String |  |
| 13 | RanchRecordId | string |  |
| 14 | Field Name | String |  |
| 15 | FieldRecordId | String |  |
| 16 | Product Category | String |  |
| 17 | Product Type | string |  |
| 18 | Weight | string | This is the pounds of seed planted |
| 19 | Vendor Name | String |  |
| 20 | VendorRecordId | String |  |
| 21 | Seed Lot Purity | String |  |
| 22 | Germination Percentage | String |  |
| 23 | Harvest Date | string | Actual start of harvesting |
| 24 | Harvest Yield | string | Actual total yield in pounds |
| 25 | Estimated Harvest Date | string | Estimated start of harvesting default=30day+datetime |
| 26 | Estimated Yield | string | Estimated total yield in poiunds |
| 27 | ItemType | String |  |
| 28 | Acres | string |  |
| 29 | Beds | String |  |
| 30 | Status | String |  |

### setProductCategories

This call creates/updates product categories as shown in the following figure. The directory name is the name coding field.



**POST:**

{webserver}/dataservice/setProductCategories/{login}/{password}/

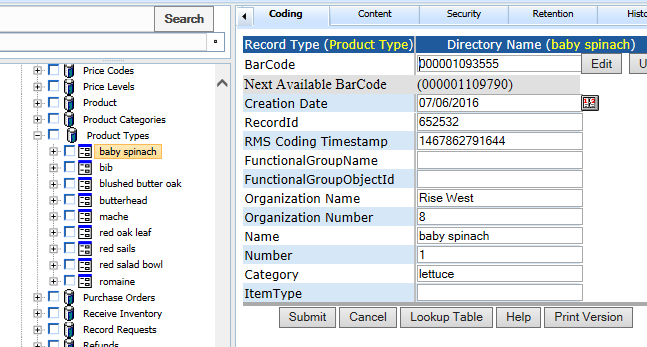
**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Name | String |  |
| 10 | Number | String |  |
| 11 | ItemType | String |  |

### setProductTypes

This call creates/updates product types as shown in the following figure. The directory name is the name coding field.



**POST:**

{webserver}/dataservice/setProductTypes/{login}/{password}/

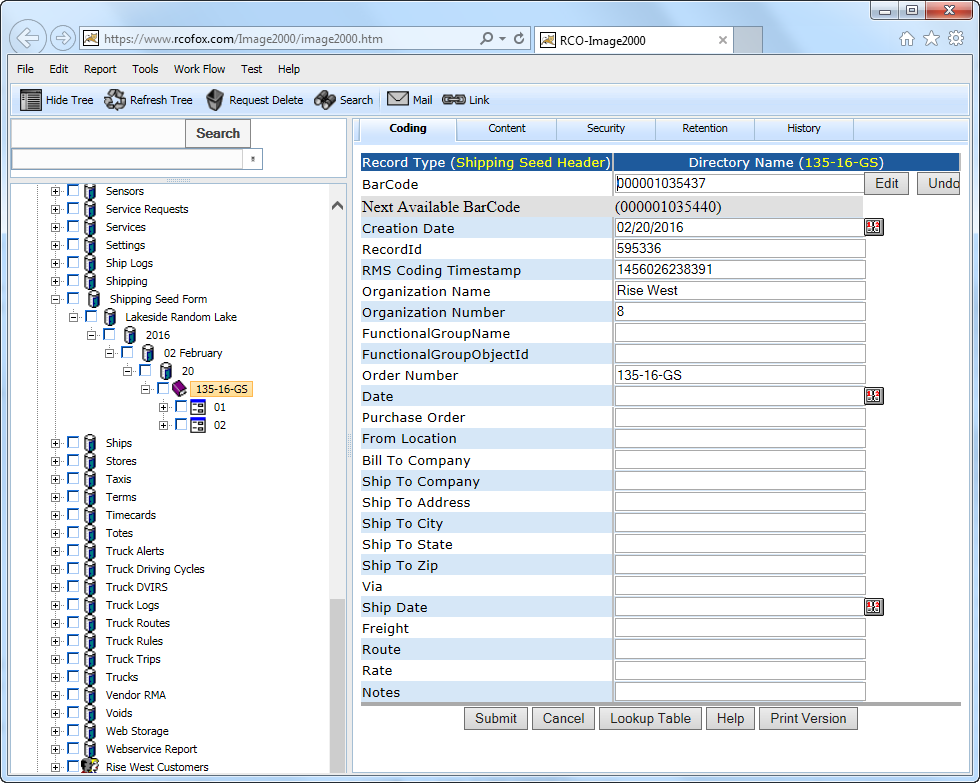
**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Name | String | Type of container {bag, box, bin, truck, pallet} |
| 10 | Number | String |  |
| 11 | ItemType | String |  |

### setShippingSeedForms

This creates/updates one or more Shipping Seed forms stored by Shipping Seed then Bill to, then year, month and day followed by the form record with a directory structure like in the following figure.



**POST:**

{webserver}/dataservice/setShippingSeedForms/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

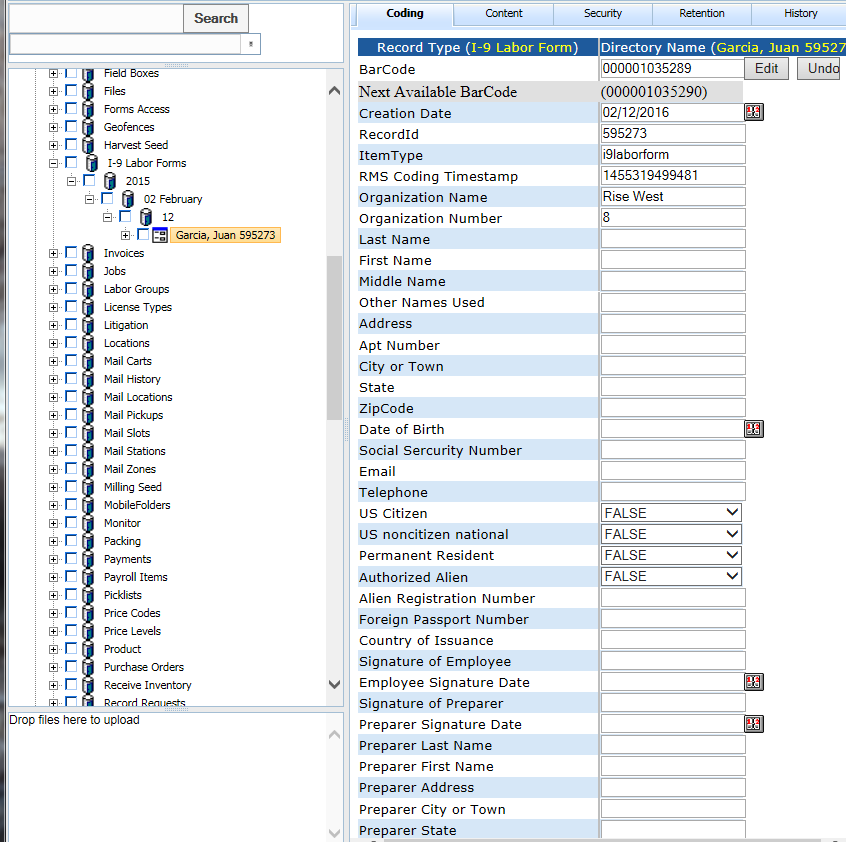
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Order Number | String |  |
| 10 | Date | String |  |
| 11 | Purchase Order | String |  |
| 12 | From Location | String |  |
| 13 | Bill To Company | String |  |
| 14 | Ship To Company | String |  |
| 15 | Ship To Address | String |  |
| 16 | Ship To City | String |  |
| 17 | Ship To State | String |  |
| 18 | Ship To Zip | String |  |
| 19 | Via | String |  |
| 20 | Ship Date | String |  |
| 21 | Freight | String |  |
| 22 | Route | String |  |
| 23 | Rate | String |  |
| 24 | Notes | String |  |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D”=this is a invoice detail item |
| 3 | objectId | string | This is the timecard detail objectId. When you do an update you must have a header. |
| 4 | objectType | string | This is the timecard detail objectType |
| 5 | MobileRecordId | string | This is the group responsible for the record |
| 6 | FunctionalGroupName | String | This is used by the local device database and is generated by deviceid+timestamp |
| 7 | organizationName | string | This is the organization name to whom the customer belongs |
| 8 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 9 | Number Sacks | string |  |
| 10 | Kind of Sacks | string |  |
| 11 | Lot Variety | string |  |
| 12 | Seed Count | string |  |
| 13 | Germination | string |  |
| 14 | TD | Date |  |
| 15 | Treatment | string |  |
| 16 | Weight | Date |  |
| 17 | Unit of Measure | string |  |
| 18 | Blend | string |  |
| 19 | State of Origin | string |  |
| 20 | Label | string |  |
| 21 | Use1 | String |  |
| 22 | Use2 | Boolean |  |
| 23 | Use3 | string |  |
| 24 | Use4 | string |  |
| 25 | Use5 | String |  |
| 26 | Weight1 | String |  |
| 27 | Weight2 | String |  |
| 28 | Weight3 | String |  |
| 29 | Weight4 | String |  |
| 30 | Weight5 | String |  |

### setI9LaborForms

This creates/updates one or more I-9 Labor Form record(s) under I-9 Labor forms first the year, month and day followed by the form record with a directory structure like in the following figure.



**POST:**

{webserver}/dataservice/setI9LaborForms/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

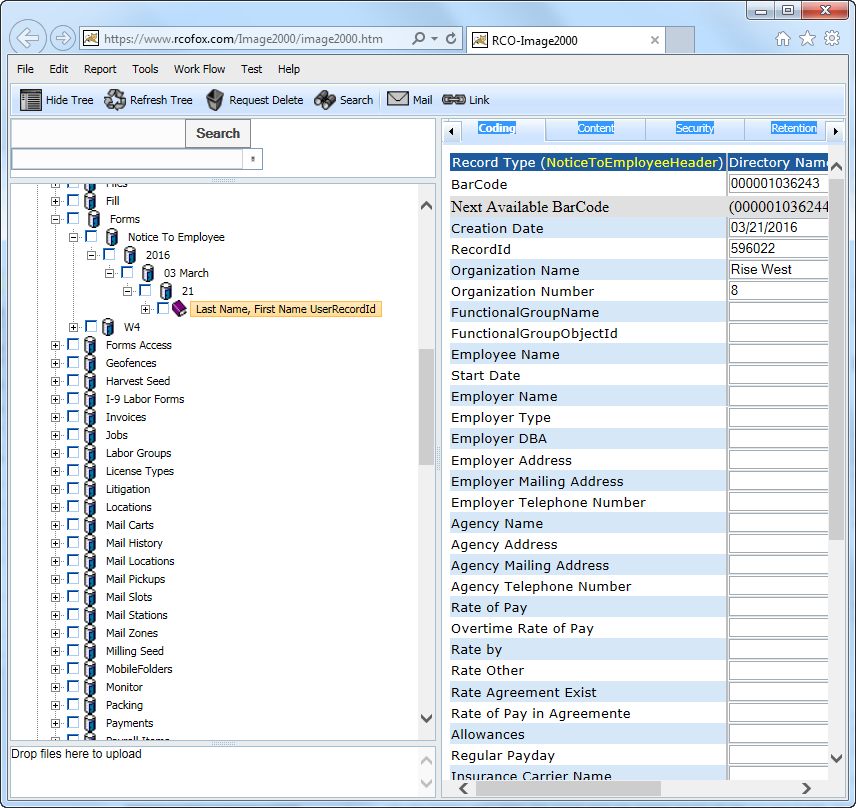
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Last Name | String |  |
| 10 | First Name | String |  |
| 11 | Middle Name | String |  |
| 12 | Other Names Used | String |  |
| 13 | Address | String |  |
| 14 | Apartment Number | String |  |
| 15 | City or Town | String |  |
| 16 | State | String |  |
| 17 | ZipCode | String |  |
| 18 | Date of Birth | String |  |
| 19 | Social Security Number | String |  |
| 20 | Email | String |  |
| 21 | Telephone | String |  |
| 22 | US Citizen | String |  |
| 23 | US noncitizen national | String |  |
| 24 | Permanent Resident | String |  |
| 25 | Authorized Alien | String |  |
| 26 | Alien Registration Number | String |  |
| 27 | Foreign Passport Number | String |  |
| 28 | Country of Issuance | String |  |
| 29 | Signature of Employee | String |  |
| 30 | Employee Signature Date | String |  |
| 31 | Signature of Preparer | String |  |
| 32 | Preparer Signature Date | String |  |
| 33 | Preparer Last Name | String |  |
| 34 | Preparer First Name | String |  |
| 35 | Preparer Address | String |  |
| 36 | Preparer City or Town | String |  |
| 37 | Preparer State | String |  |
| 38 | Preparer Zip Code | String |  |
| 39 | List A 1.1 Document Title | String |  |
| 40 | List A 1.2 Issuing Authority | String |  |
| 41 | List A 1.3 Document Number | String |  |
| 42 | List A 1.4 Expiration Date | Date |  |
| 43 | List A 2.1 Document Title | String |  |
| 44 | List A 2.2 Issuing Authority | String |  |
| 45 | List A 2.3 Document Number | String |  |
| 46 | List A 2.4 Expiration Date | Date |  |
| 47 | List A 3.1 Document Title | String |  |
| 48 | List A 3.2 Issuing Authority | String |  |
| 49 | List A 3.3 Document Number | String |  |
| 50 | List A 3.4 Expiration Date | Date |  |
| 51 | List B 1.1 Document Title | String |  |
| 52 | List B 1.2 Issuing Authority | String |  |
| 53 | List B 1.3 Document Number | String |  |
| 54 | List B 1.4 Expiration Date | Date |  |
| 55 | List C 1.1 Document Title | String |  |
| 56 | List C 1.2 Issuing Authority | String |  |
| 57 | List C 1.3 Document Number | String |  |
| 58 | List C 1.4 Expiration Date | Date |  |
| 59 | Signature of Employer |  |  |
| 60 | Employer Signature Date | Date |  |
| 61 | Title of Employer | String |  |
| 62 | Employer Last Name | String |  |
| 63 | Employer First Name | String |  |
| 64 | Employer Business Name | String |  |
| 65 | Employer Business Address | String |  |
| 66 | Employer City or Town | String |  |
| 67 | Employer State | String |  |
| 68 | Employer Zip Code | String |  |
| 69 | Employees first day of employment | Date |  |
| 70 | Rehire Last Name | String |  |
| 71 | Rehire First Name | String |  |
| 72 | Rehire Middle Name | Date |  |
| 73 | Rehire Date | Date |  |
| 74 | Rehire Document Title | String |  |
| 75 | Rehire Document Number | String |  |
| 76 | Rehire Expiration Date | Date |  |
| 77 | Rehire Signature of Employer | String |  |
| 78 | Rehire Employer Signature Date | Date |  |
| 79 | Rehire Employer Print Name | String |  |
| 80 | ItemType | string |  |
| 81 | Date Received | Date | This date should never change or be edited |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D”=this is a invoice detail item |
| 3 | objectId | string | This is the timecard detail objectId. When you do an update you must have a header. |
| 4 | objectType | string | This is the timecard detail objectType |
| 5 | MobileRecordId | string | This is the group responsible for the record |
| 6 | FunctionalGroupName | String | This is used by the local device database and is generated by deviceid+timestamp |
| 7 | organizationName | string | This is the organization name to whom the customer belongs |
| 8 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 9 | Document Title | string |  |
| 10 | Issuing Authority | string |  |
| 11 | Expiration Date | Date |  |
| 12 | Signature Date | Date |  |
| 13 | Signature Name | String |  |
| 14 | Description | string |  |
| 15 | ItemType | string |  |
| 16 | Document Type | String |  |
| 17 | Document Date | Date |  |
| 18 | Reviewed By | String |  |
| 19 | Reviewed Date | Date |  |

### setNoticeToEmployeeForms

This creates/updates one or more Notice to Employee Form record(s) under Forms then Notice To Employee then the year, month and day followed by the form record with a directory structure like in the following figure.



**POST:**

{webserver}/dataservice/setNoticeToEmployeeForms/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

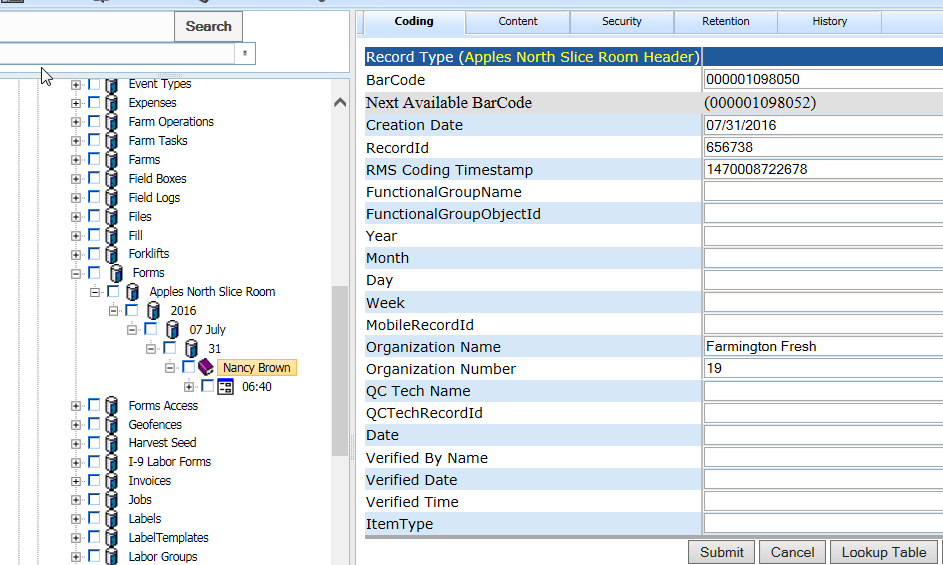
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Employee Name | String |  |
| 10 | Start Date | String |  |
| 11 | Employer Name | String |  |
| 12 | Employer Type | String |  |
| 13 | Employer DBA | String |  |
| 14 | Employer Address | String |  |
| 15 | Employer Mailing Address | String |  |
| 16 | Employer Telephone Number | String |  |
| 17 | Agency Name | String |  |
| 18 | Agency Address | String |  |
| 19 | Agency Mailing Address | String |  |
| 20 | Agency Telephone Number | String |  |
| 21 | Rate of Pay | String |  |
| 22 | Overtime Rate of Pay | String |  |
| 23 | Rate by | String |  |
| 24 | Rate Other | String |  |
| 25 | Rate Agreement Exist | String |  |
| 26 | Rate of Pay in Agreement | String |  |
| 27 | Allowances | String |  |
| 28 | Regular Payday | String |  |
| 29 | Insurance Carrier Name | String |  |
| 30 | Insurance Carrier Address | String |  |
| 31 | Insurance Carrier Telphone Number | String |  |
| 32 | Insurance Carrier Policy Number | String |  |
| 33 | Self Insured Checkbox | String |  |
| 34 | Self Insured Certificate Number | String |  |
| 35 | Sick Leave Checkbox 1 | String |  |
| 36 | Sick Leave Checkbox 2 | String |  |
| 37 | Sick Leave Checkbox 3 | String |  |
| 38 | Sick Leave Checkbox 4 | String |  |
| 39 | Employer Representative Name | String |  |
| 40 | Employer Representative Signature | String |  |
| 41 | Employer Signature Date | String |  |
| 42 | Employee Full Name | Date |  |
| 43 | Employee Signature | String |  |
| 44 | Employee Signature Date | String |  |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D”=this is a invoice detail item |
| 3 | objectId | string | This is the timecard detail objectId. When you do an update you must have a header. |
| 4 | objectType | string | This is the timecard detail objectType |
| 5 | MobileRecordId | string | This is the group responsible for the record |
| 6 | FunctionalGroupName | String | This is used by the local device database and is generated by deviceid+timestamp |
| 7 | organizationName | string | This is the organization name to whom the customer belongs |
| 8 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 9 | Signature Date | Date |  |
| 10 | Signature Name | String |  |
| 11 | Document Title | String |  |
| 12 | Document Type | String |  |

### setApplesNorthSliceRoomForms

This creates/updates one or more Apples North Slice Room Form record(s) under Forms then Apples North Slice Room then the year, month and day followed by the form record with a directory structure like in the following figure. The header record is named by the QC Tech Name and the detail record is named with the time.



**POST:**

{webserver}/dataservice/setApplesNorthSliceRoomForms/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

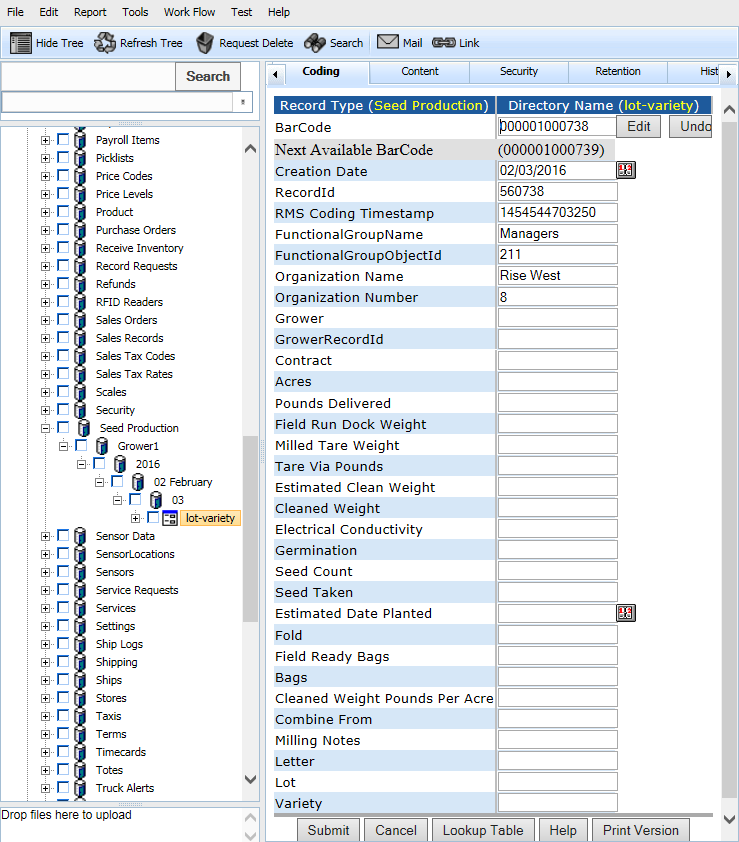
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | QC Tech Name | String |  |
| 10 | QCTechRecordId | String |  |
| 11 | Date | String |  |
| 12 | Verified By Name | String |  |
| 13 | Verified DateTime | String |  |
| 15 | ItemType | String | ApplesNorthSliceRoomHeader |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D”=this is a invoice detail item |
| 3 | objectId | string | This is the timecard detail objectId. When you do an update you must have a header. |
| 4 | objectType | string | This is the timecard detail objectType |
| 5 | MobileRecordId | string | This is the group responsible for the record |
| 6 | FunctionalGroupName | String | This is used by the local device database and is generated by deviceid+timestamp |
| 7 | organizationName | string | This is the organization name to whom the customer belongs |
| 8 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 9 | DateTime | String | Server will split into date and time |
| 10 | Package | String |  |
| 11 | Bin Dump | String |  |
| 12 | Atlas Tank | String |  |
| 13 | Slice Flume | String |  |
| 14 | CA Tank | String |  |
| 15 | Fe-5.0 | String |  |
| 16 | Non Fe-3.0 | String |  |
| 17 | 5.0 304 SS | String |  |
| 18 | 5.0 316 SS | String |  |
| 19 | Detector ID Tag Checked Date | Date |  |
| 20 | Monitored By | String |  |
| 21 | Corrective Actions | String |  |
| 22 | Notes | String |  |

### setSeedProduction

This creates a seed production record under Seed Production first the grower then year, month and day with a directory structure like in the following figure.



**POST:**

{webserver}/dataservice/setSeedProduction/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Grower |  |  |
| 10 | GrowerRecordId | String |  |
| 11 | Contract | String |  |
| 12 | Acres | Number |  |
| 13 | Pounds Delivered | Number |  |
| 14 | Field Run Dock Weight | String |  |
| 15 | Milled Tare Weight | Number |  |
| 16 | Tare via Pounds | Number |  |
| 17 | Estimated Clean Weight | Number |  |
| 18 | Cleaned Weight | Number |  |
| 19 | Electrical Conductivity | Float |  |
| 20 | Germination | String |  |
| 21 | Seed Count | Number |  |
| 22 | Seed Taken | Number |  |
| 23 | Estimated Date Planted | Date |  |
| 24 | Lot Used for SS | String |  |
| 25 | Fold | String |  |
| 26 | Field Reay Bags | String |  |
| 27 | Bags | String |  |
| 28 | Cleaned Weight Pounds Per Acre | Number |  |
| 29 | Combine From | String |  |
| 30 | Milling Notes | String |  |
| 31 | Letter | String |  |
| 32 | Lot | String |  |
| 33 | Variety | String |  |
| 34 | Area Inspected | Boolean |  |
| 35 | Field Inspected | Boolean |  |

### setSensorData

This call generates data in the tree every 30 minutes with the following directory structure (grower, then ranch, then field, then sensor type, then sensor name + sensorrecordid then year, month, day and finally time. The record type is sensor data.



Using the SensorRecordId the rms will copy the following coding fields from the sensor

Latitude

Longitude

Item Number

Description

Manufacturer

Sensor Alert Email

Sensor Alert SMS

Sensor High Limit

Sensor Low Limit

Grower

GrowerRecordId

Ranch

RanchRecordId

Field

FieldRecordId

**POST:**

{webserver}/dataservice/setSensorData/{login}/{password}/

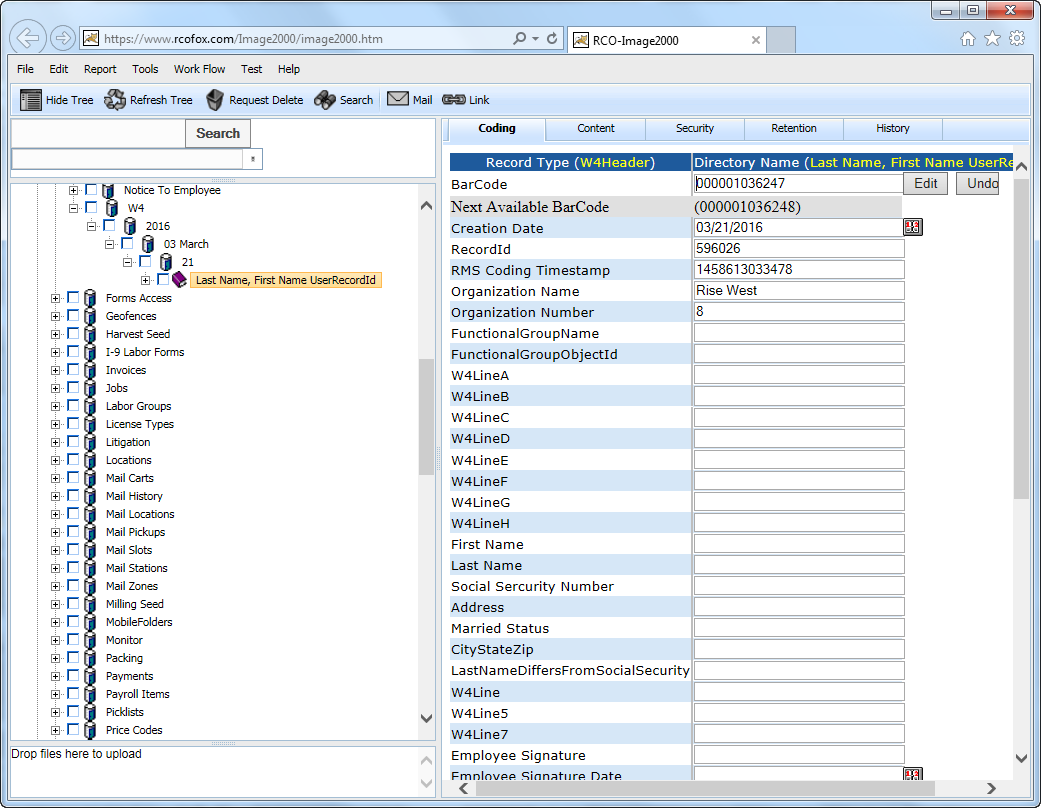
**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 7 | Organization Name | String |  |
| 7 | Organization Name | String |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Value | Float | Metric units |
| 10 | DateTime | String |  |
| 11 | Sensor Name | String |  |
| 12 | SensorRecordId | String |  |

### setW4Forms

This creates/updates one or more W4 Form record(s) under forms/w4 then the year, month and day followed by the form record with a directory structure like in the following figure.



**POST:**

{webserver}/dataservice/setW4Forms/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | W4LineA | String |  |
| 10 | W4LineB | String |  |
| 11 | W4LineC | String |  |
| 12 | W4LineD | String |  |
| 13 | W4LineE | String |  |
| 14 | W4LineF | String |  |
| 15 | W4LineG | String |  |
| 16 | W4LineH | String |  |
| 17 | First Name | String |  |
| 18 | Last Name | String |  |
| 19 | Social Security Number | String |  |
| 20 | Address | String |  |
| 21 | Married Status | String |  |
| 22 | CityStateZip | String |  |
| 23 | LastNameDiffersFromSocialSecurity | String |  |
| 24 | W4Line5 | String |  |
| 25 | W4Line6 | String |  |
| 26 | W4Line7 | String |  |
| 27 | EmployeeSignature | String |  |
| 28 | EmployeeSignatureDate | String |  |
| 29 | EmployerNameAndAddress | String |  |
| 30 | OfficeCode | String |  |
| 31 | Employer Identification Number | String |  |
| 32 | W4DeductionsLine1 | String |  |
| 33 | W4DeductionsLine2 | String |  |
| 34 | W4DeductionsLine3 | String |  |
| 35 | W4DeductionsLine4 | String |  |
| 36 | W4DeductionsLine5 | String |  |
| 37 | W4DeductionsLine6 | String |  |
| 38 | W4DeductionsLine7 | String |  |
| 39 | W4DeductionsLine8 | String |  |
| 40 | W4DeductionsLine9 | String |  |
| 41 | W4DeductionsLine10 | String |  |
| 42 | W4MultipleJobsLine1 | String |  |
| 43 | W4MultipleJobsLine2 | String |  |
| 44 | W4MultipleJobsLine3 | Date |  |
| 45 | W4MultipleJobsLine4 | String |  |
| 46 | W4MultipleJobsLine5 | String |  |
| 47 | W4MultipleJobsLine6 | String |  |
| 48 | W4MultipleJobsLine7 | Date |  |
| 49 | W4MultipleJobsLine8 | String |  |
| 50 | W4MultipleJobsLine9 | String |  |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D”=this is a invoice detail item |
| 3 | objectId | string | This is the timecard detail objectId. When you do an update you must have a header. |
| 4 | objectType | string | This is the timecard detail objectType |
| 5 | MobileRecordId | string | This is the group responsible for the record |
| 6 | FunctionalGroupName | String | This is used by the local device database and is generated by deviceid+timestamp |
| 7 | organizationName | string | This is the organization name to whom the customer belongs |
| 8 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 12 | Signature Date | Date |  |
| 13 | Signature Name | String |  |
| 14 | Document Title | String |  |
| 15 | Document Type | String |  |

### setSensors

This creates/edits a record type called sensor. This is primarily used to setup a sensor. We no longer use the content of the sensor but have a separate sql table with special calls. The content of the record contains the file describing the sensor data (Examplle: date, time, sensor type, value, latitude, longitude). Note that the actual sensors need to get added to the accounting system as a non-inventory part with tracking equals to true if serialized. For the directory structure you have the following.

Sensors

Ranch

Sensor Type

Sensor Name

**POST:**

{webserver}/dataservice/setSensors/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

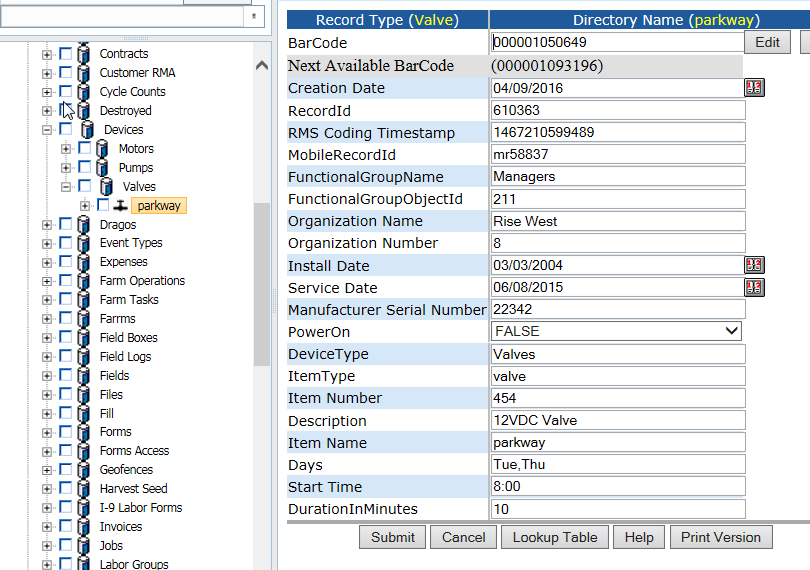
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | mobilerecordid | string | If invoice header exists then this is Invoice Header |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Organization Name | string |  |
| 8 | Organization Number | string |  |
| 9 | ItemType | string |  |
| 10 | SensorType | date |  |
| 11 | Sensor Name | date |  |
| 12 | Vendor Name | date |  |
| 13 | Manufacturer Serial Number | date |  |
| 14 | Latitude | string |  |
| 15 | Longitude | string |  |
| 16 | Item Number | string | This is the accounting item number for the sensor |
| 17 | Description | string | This is the description of the sensor |
| 18 | Date | string | Installation Date YYYYMMDD |
| 19 | Last Worked | string | Date sensor was last services YYYYMMDD |
| 20 | Last Read | string | Date Time YYYYMMDD HH:MM:SS.SSS |
| 21 | Manufacturer | string | This is the preferred vendor of this sensor |
| 22 | Value |  |  |
| 23 | Alert |  |  |
| 24 | Sensor Alert Email |  |  |
| 25 | Sensor Alert SMS |  |  |
| 26 | Sensor High Limit |  |  |
| 27 | Sensor Low Limit |  |  |
| 28 | Grower |  |  |
| 29 | GrowerRecordId |  |  |
| 30 | Ranch |  |  |
| 31 | RanchRecordId |  |  |
| 32 | Field |  |  |
| 33 | FieldRecordId |  |  |

## Device Service

The following services are used to control physical devices.

### setValves

This creates/updates one or more valves under organization then Devices then devicetype and then the device record which is named in the directory item number description manufacturer serial number with a directory structure like in the following figure.



**POST:**

{webserver}/deviceservice/setValves/{login}/{password}/

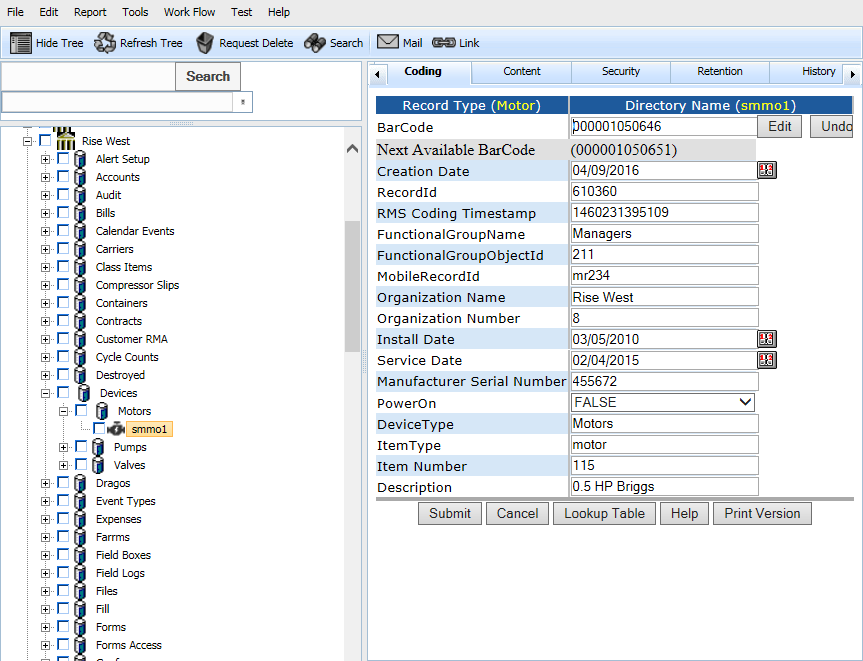
**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Install Date | Date |  |
| 10 | Service Date | Date |  |
| 11 | Manufacturer Serial Number | String |  |
| 12 | PowerOn | Boolean |  |
| 13 | DeviceType | String |  |
| 14 | ItemType | String |  |
| 15 | Item Number | String |  |
| 16 | Description | String |  |
| 17 | Item Name | String |  |
| 18 | Days | String |  |
| 19 | Start Time | String |  |
| 20 | DurationInMinutes | String |  |
| 21 | Latitude | String |  |
| 22 | Longitude | String |  |
| 23 | Grower | String |  |
| 24 | GrowerRecordId | String |  |
| 25 | Ranch | string |  |
| 26 | RanchRecordId | String |  |
| 27 | Field Name | String |  |
| 28 | FieldRecordId | String |  |

### setPumps

This creates/updates one or more valves under organization then Devices then devicetype and then the device record which is named in the directory item number description manufacturer serial number with a directory structure like in the following figure.



**POST:**

{webserver}/deviceservice/setPumps/{login}/{password}/

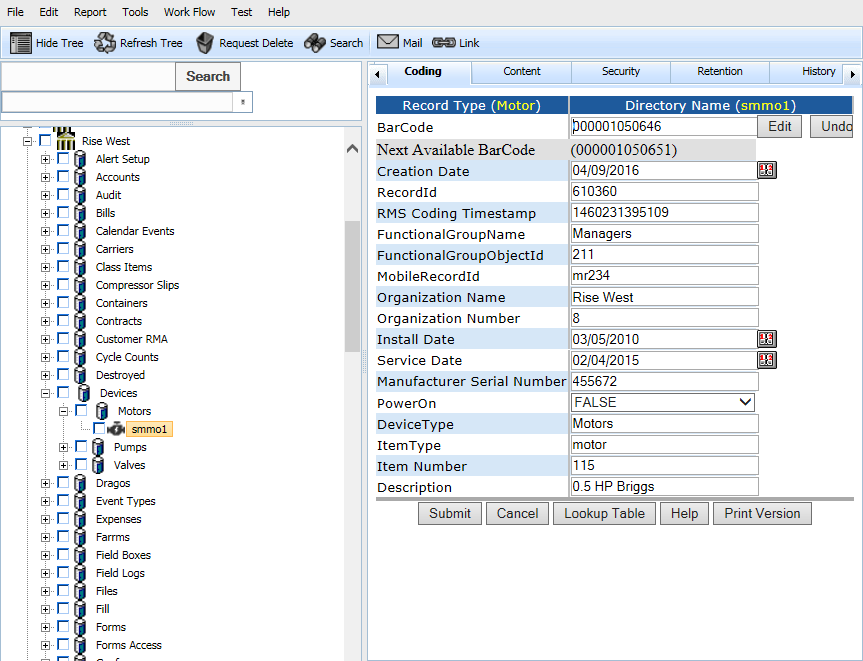
**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Install Date | Date |  |
| 10 | Service Date | Date |  |
| 11 | Manufacturer Serial Number | String |  |
| 12 | PowerOn | Boolean |  |
| 13 | DeviceType | String |  |
| 14 | ItemType | String |  |
| 15 | Item Number | String |  |
| 16 | Description | String |  |
| 17 | Item Name | String |  |
| 18 | Days | String |  |
| 19 | Start Time | String |  |
| 20 | DurationInMinutes | String |  |
| 21 | Latitude | String |  |
| 22 | Longitude | String |  |
| 23 | Grower | String |  |
| 24 | GrowerRecordId | String |  |
| 25 | Ranch | string |  |
| 26 | RanchRecordId | String |  |
| 27 | Field Name | String |  |
| 28 | FieldRecordId | String |  |

### setMotors

This creates/updates one or more motors under organization then Devices then devicetype and then the device record which is named in the directory item number description manufacturer serial number with a directory structure like in the following figure.



**POST:**

{webserver}/deviceservice/setMotors/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | ObjectId | string | This is the sensor objectid |
| 4 | ObjectType | string | This is the sensor objecttype |
| 5 | MobileRecordId | string |  |
| 6 | Functional Group Name | string | This is security for login user what they can see. Optional unles you want security and then you use this and the org number |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Install Date | Date |  |
| 10 | Service Date | Date |  |
| 11 | Manufacturer Serial Number | String |  |
| 12 | PowerOn | Boolean |  |
| 13 | DeviceType | String |  |
| 14 | ItemType | String |  |
| 15 | Item Number | String |  |
| 16 | Description | String |  |
| 17 | Item Name | string |  |
| 18 | Days | String |  |
| 19 | Start Time | String |  |
| 20 | DurationInMinutes | String |  |
| 21 | Latitude | String |  |
| 22 | Longitude | String |  |
| 23 | Grower | String |  |
| 24 | GrowerRecordId | String |  |
| 25 | Ranch | string |  |
| 26 | RanchRecordId | String |  |
| 27 | Field Name | String |  |
| 28 | FieldRecordId | String |  |

## Directory Service

The restful directory service interfaces are created based on directory APIs defined in section 6.1.1, which allows applications to modify the rms directory structure.

Note that some of the restful interfaces/APIs such as getRecordCount, getRecordsIdsAll, and getRecordIdsUpdated are described in Qucikbook Service section. Some of the restful APIs will be moved around between directory service and quickbook service as the implementation of backend APIs is finalized.

### createRecordInDirectory

**Description:**

This restful interface is used to create objects on the tree.

**POST:**

{webserver}/directoryservice/createRecordInDirectory/{loginId}/{password}/{parentTreeId}/{displayType}/{objectName}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| parentTreeId | TreeId of the parent where you want to create your record. |
| displayType | Record type (eg. part, service, user) |
| objectName | Name of the record that appears in directory |

### deleteDirectoryBranch

This function deletes a given directory tree id and all children and all nodes point to this record.

**POST:**

{webserver}/directoryservice/deleteDirectoryBranch /{loginId}/{password}/{TreeId}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getDirectoryIds - action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| treeId | This is the directory node where you want to start the deletion |

### getAllChildrenList

Get the children, as a list (Vector) of NodeListItems, of a tree node identified by its tree id.  Tree security (Visible, View, List, Block Inheritance) will be employed. Children of root node are retrieved by passing in treeId = 0.

**GET:**

{webserver}/directoryservice/getAllChildrenList/{loginId}/{password}/{treeId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| treeId | tree id of tree node whose children are being requested. Pass 0 for children of root node. |

### getChildrenByTreeId

Get the children, as a list (Vector) of NodeListItems, of a tree node identified by its tree id.  Tree security (Visible, View, List, Block Inheritance) will be employed. Children of root node are retrieved by passing in treeId = 0.

**GET:**

{webserver}/directoryservice/getChildrenByTreeId/{loginId}/{password}/{treeId}/{displayType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| treeId | tree id of tree node whose children are being requested. Pass 0 for children of root node. |
| displayType | Optional parameter for display type of children records example eFile. |

### getChildrenDirectoryIds

This restful interface is used to return all children of a particular Display Type.

**GET:**

{webserver}/directoryservice/getChildrenDirectoryIds/{loginId}/{password}/{parentobjectId}/{parentobjectType}/{childDisplayType}

**EXAMPLE:**

{webserver}/directoryservice/getChildrenDirectoryIds/login/password//2/NRT204/Item Receipt Detail

**RESPONSE:**

[{"treeId":357674,"objectId":2,"objectType":"NRT205","name":"906-103 MDT Touch Terminal"},{"treeId":357675,"objectId":3,"objectType":"NRT205","name":"906-107 MDT Mounting Framel"}]OK (200) - OK

**ARGUMENTS:**

no empty arguments allowed

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getDirectoryIds - action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Parent Object Id | This is the directory node parent record |
| Parent Object Type | This is the record type of the parent node |
| ChildDisplyType | This is the object type of the detail record Example: Item Receipt Detail |

### getDirectoryId

This restful interface is used to get directory parent information.

**GET:**

{webserver}/directoryservice/getDirectoryId/{loginId}/{password}/{displayType}/{objectName}

Note that securityCode = organizationNumber

**EXAMPLE:**

{webserver}/directoryservice/getDirectoryId/login/plogin/User Group/aName

**ARGUMENTS:**

no empty arguments allowed

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getDirectoryIds - action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| displayType | This is the record type (eg. user, part, service) |
| objectName | Name of the directory record |

### getDirectoryIds

This restful interface is used to get ids and time stamps for a given displayType under a directory node. The parentTreeId is obtained from getDirectoryId. This call returns id structures which include timestamp information.

**GET:**

{webserver}/directoryservice/getDirectoryIds/{loginId}/{password}/{displayType}/{parentTreeId}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getDirectoryIds - action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| objectId | Record id for what you want to delete |
| objectType | Record type for what you want to delete |

### getEfileDownloadPath

This is used to get a url path to the file for a given objectid and objecttype. The function returns the string url path. What this function does right now is it copies the data for storage to a download directory and decrypts the data.

**GET:**

{webserver}/directoryservice/getEfileDownloadPath/{loginId}/{password}/{key}/{objectId}/{objecType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getDirectoryIds - action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Key | Encryption key. For now same as password |
| objectId | This is the objectid of the record you want |
| objectType | This is the objecttype of the record you want |

### getFileFolders

This call returns a list of folders (objectid, objecttype) that contain one or more files that the login user has access to. When an eFile is created filter.Document Number (filename) and filter.Parth.

**GET:**

{webserver}/directoryservice/getFileFolders/{loginId}/{password}/{MaxLevels}/{isTreeSecurity}/{minLevels}/{stopFolder}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getDirectoryIds - action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| MaxLevels | How many levels from the leaf upward do you want to see optional number, the maximum number of folders that will be included in the path. May result in non-unique paths. |
| isTreeSecurity | BOOLEAN TRUE/FALSE checks whether the user has visibility on the parent folder of the records optional, if "true", applies an additional tree security check on the queried paths. |
| minLevels | optional number, the minimum number of folders to be included in the path, not exceeding the maximum number of folders, and not exceeding the actual number of folder levels, and not exceeding the number of folders up to the "stopFolder" folder. |
| stopFolder | optional string, the folder name of an ancestor folder which will not be included in the path along with its ancestors. |

\* **@return** - returns a list of 2-element lists of strings. Each two-element list contains the Tree Id of the lowest folder in the path, and the second element contains a combination of folder names which tries to form a unique path identifier, subject to the effects of the input parameters.

\* **@throws** Exception

### getFilterCategoryPaths

This call returns a list of paths that the user has efiles under.

**GET:**

{webserver}/directoryservice/getFilterCategoryPaths/{loginId}/{password}/{RecordType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getDirectoryIds - action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| RecordType | Optional with default = all records types that have puiblishuserrecordid |

### getContainerPaths

This call returns a list of the parent records of a given record type. As an example suppose you want to get a list of all the parent containers for the search reports. For the record type = Search this will return a list of ReportStorage containers. You then need to get the children of a selected parent node. This can also be used to get all the parent nodes of say a legal document record type. In contrast to getNestList which walk down the tree getContainerPaths walks up the tree.

**GET:**

{webserver}/directoryservice/getContainerPaths/{loginId}/{password}/{RecordType}/{maxLevels}/{minLevels}/{isTreeSecurity}/{stopFolder}/

Returns

TreeId, ObjectId, ObjectType, DirectoryNodeName

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| RecordType | Recordtype you want to get |
| MaxLevels | How many levels from the leaf upward do you want to see optional number, the maximum number of folders that will be included in the path. May result in non-unique paths. |
| isTreeSecurity | BOOLEAN TRUE/FALSE checks whether the user has visibility on the parent folder of the records optional, if "true", applies an additional tree security check on the queried paths. |
| minLevels | optional number, the minimum number of folders to be included in the path, not exceeding the maximum number of folders, and not exceeding the actual number of folder levels, and not exceeding the number of folders up to the "stopFolder" folder. |
| stopFolder | optional string, the folder name of an ancestor folder which will not be included in the path along with its ancestors. |

\* **@return** - returns a list of 2-element lists of strings. Each two-element list contains the Tree Id of the lowest folder in the path, and the second element contains a combination of folder names which tries to form a unique path identifier, subject to the effects of the input parameters.

\* **@throws** Exception

### getNodeInfoByMobileRecordId

This returns the node information structure that includes encryptMode, objectId, objectType for a given user.

**GET:**

{webserver}/directoryservice/getNodeInfoByMobileRecordId/{loginId}/{password}/{/MobileRecordId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| MobileRecordId | RecordId |

### getNodeInfoByObjectIdObjectType

This returns the node information structure that includes encryptMode for a given user.

**GET:**

{webserver}/directoryservice/getNodeInfoByObjectdObjectType/{loginId}/{password}/{/ObjectId}/{/ObjectType}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| ObjectId | ObjectId |
| ObjectType | ObjectType |

### getNodeInfoByRecordId

This returns the node information structure that includes encryptMode, objectId, objectType for a given user.

**GET:**

{webserver}/directoryservice/getNodeInfoByRecordId/{loginId}/{password}/{/RecordId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| RecordId | RecordId |

### getNodeInfoByTreeId

This returns the node information structure that includes objectId and objectType for a give user.

**GET:**

{webserver}/directoryservice/getNodeInfoByTreeId/{loginId}/{password}/{/TreeId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| TreeId | Directory tree id |

### getNodesByItemType

Gets the ids and time stamps for a given displayType under a directory node. The parentTreeId is obtained from getDirectoryId.

**GET:**

{webserver}/directoryservice/getNodesByItemType/{loginId}/{password}/{itemType}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| itemType | Category, Subject, Topic |

### getNodesByRecordType

Gets the nodeinfo structure for all directory nodes of a given record type for the given login of their organization. This call should only be used for a small number of records like names of forms or report storage containers.

**GET:**

{webserver}/directoryservice/getNodesByRecordType/{loginId}/{password}/{RecordType}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| RecordType | Example Search |

### getChildNodes

Returns the nodeInfo from a starting parent tree id.

**GET:**

{webserver}/directoryservice/getChileNodes/{loginId}/{password}/{parentTreeId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getDirectoryIds - action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| ParentTreeId | Starting parent tree id. Use 0 to start at the top |

### getEncryptMode

Returns the encrypt mode for a given RmsDocumentNumber.

0 = no encryption

1 = AES 256

2 = Adobe PDF encryption

5 =

**GET:**

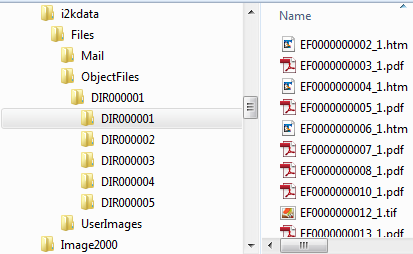
{webserver}/directoryservice/getEncryptMode/{loginId}/{password}/{RmsDocumentNumber}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getDirectoryIds - action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| RmsDocumentNumber | Unique file identifier |

### readWebFile

Reads a file from the RMS web server from the i2kdata folder (whose path is specified in the serverProperties.txt file. The function returns a string “Success” if the read is successful or a string explaining why the call failed.



**GET:**

{webserver}/directoryservice/readWebFile/{loginId}/{password}/{RmsDocumentNumber}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getDirectoryIds - action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| RmsDocumentNumber | Unique file identifier |

### setDirectoryNodeName

This will set a directory node name using the node’s recordId or treeId or objectId, objectType. The call will return OK if the function is successful.

**GET:**

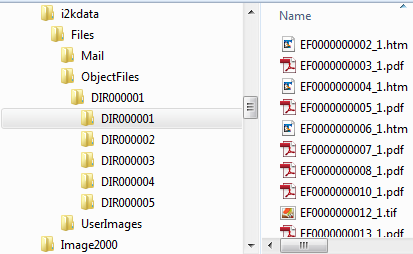
{webserver}/directoryservice/setNodeNameByRecordId/{loginId}/{password}/{recordId}/{NewNodeName}/{treeId}/{objectId}/{objectType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| recordId | This is the recordId of the directory node you want to change |
| NewNodeName | This is the new directory node name |
| treeId | Directory node treeId |
| objectId | Directory node objectId |
| objectType | Directory node objectType |

### writeWebFile

Write a file to the RMS web server in the i2kdata folder (whose path is specified in the serverProperties.txt file. The function returns a string “Success” if the write is successful or a string explaining why the call failed.



**GET:**

{webserver}/directoryservice/writeWebFile/{loginId}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getDirectoryIds - action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

## Email Service

The following calls deal with sending or receiving emails with or without attachments.

### EmailFileToList

Send an attachment to every email in the attached csv file. Return a count of all the emails that were received.

**POST:**

{webserver}/emailservice/emailFileToList/{login}/{password}/{serverFilename}/{Subject}/{fromEmailAddress}

**Returns:**

N – number of emails sent

**CSV:**

“emailAddress”

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| serverFilename | Complete file path to the text file that will be attached to email |
| Subject | Subject line in the email |
| fromEmailAddress | Set the from email address |

### EmailMessageToFunctionalGroup

Send an email message to all the members of a functional group. Return a count of all the emails that were received.

**POST:**

{webserver}/emailservice/emailMessageToFunctionalGroup/{login}/{password}/{message}/{functional group name}/{Subject}/{fromEmailAddress}

**Returns:**

N – number of emails sent successfully

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Message | String that you want to email all members of the functional group |
| Functional Group Name | Name of the functional group members that you want to send a message to |
| Subject | Subject line in the email |
| fromEmailAddress | Set the from email address |

### SendEmail

Send file attachment to email address.

**POST:**

{webserver}/emailservice/SendEmail/{login}/{password}/{To}/{Subject}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| To | To email address |
| Subject | Subject line in the email |

## Forms Service

### setBoxes

This call creates/edits/deletes/updates central file records. The directory structure is the following

Boxes

Department

YYYY

MM

DD

Box Number Title

**POST:**

{webserver}/formsservice/setBoxes/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | string |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Record Number | String |  |
| 10 | OCWD Box Number | String |  |
| 11 | Iron Mountain Box Number | String |  |
| 12 | Record Series Code | String |  |
| 13 | Record Series Title | String |  |
| 14 | Description | String |  |
| 15 | Media | String |  |
| 16 | Department | String |  |
| 17 | ContactDepartment | String |  |
| 18 | From Date | Date |  |
| 19 | Up to Date | Date |  |
| 20 | Retention Period | String |  |
| 21 | Retention Exception | String |  |
| 22 | Destruction Date | Date |  |
| 23 | Destroyed Date | Date |  |
| 24 | Notes | String |  |
| 25 | Checked Out First Name | String |  |
| 26 | Checked Out Last Name | String |  |
| 27 | Check Out Date | string |  |
| 28 | Status | string |  |
| 29 | Date Stamp | String |  |
| 30 | Last Updated | String |  |
| 31 | Record Date | Date |  |
| 32 | Destruction Hold | Boolean |  |
| 33 | Site Name | String |  |
| 34 | Site Number | String |  |
| 35 | Aisle | String |  |
| 36 | Bay | String |  |
| 37 | Shelf | String |  |
| 38 | Shelf Location | String |  |

### setCentralFiles

This call creates/edits/deletes/updates central file records. The directory structure is the following

OfficeOfRecord

YYYY

MM

DD

FileTitle

**POST:**

{webserver}/formsservice/setCentralFiles/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | string |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Central File Record Number | Number |  |
| 10 | Class | String |  |
| 11 | File Class | String |  |
| 12 | File Title | String |  |
| 13 | Notes | String |  |
| 14 | ACode | String |  |
| 15 | OfficeOfRecord | String |  |
| 16 | NCD1 | String |  |
| 17 | NCD2 | String |  |
| 18 | Star Update | String |  |
| 19 | Date Stamp | String |  |
| 20 | Date | Date |  |
| 21 | Record Series Code | String |  |
| 22 | NCD3 | String |  |
| 23 | NCD4 | String |  |
| 24 | Checked Out To | String |  |
| 25 | CapitalProjectNum | String |  |
| 26 | Record Series Number | String |  |
| 27 | From Date | Date |  |
| 28 | Up To Date | Date |  |
| 29 | TotalRetention | String |  |
| 30 | Destruction Date | Date |  |
| 31 | Retention Exception | String |  |
| 32 | Follow Up | String |  |
| 33 | Contract Number | String |  |
| 34 | Checked Out Date | Date |  |
| 35 | Date Destroyed | Date |  |
| 36 | Actions | String |  |

### setEngineeringDrawings

This call creates/edits/deletes/updates engineering drawing records. The directory structure is the following

Name of Project

YYYY

MM

DD

Contract Number Designer

**POST:**

{webserver}/formsservice/setEngineeringDrawing/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | string |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Name of Project | Number |  |
| 10 | Contract Number | String |  |
| 11 | Designer | String |  |
| 12 | Date | String |  |
| 13 | Tif Image | String |  |
| 14 | Pdf Image | String |  |
| 15 | Record Series Code | String |  |

### setLibraryMaterials

This call creates/edits/deletes/updates library material records. The directory structure is the following where all library material is stored under a directory node called Library (storage record type). The library material is divided by author and then the date of the material. The data record is of record type “Library Material” and the directory node name is the call number followed by up to 40 characters of the title.

Library

Author

YYYY

MM

DD

CallNumber Title

**POST:**

{webserver}/formsservice/setLibraryMaterials/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | string |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Abstract | String |  |
| 10 | Author | String |  |
| 11 | Borrower | String |  |
| 12 | Call Number | String |  |
| 13 | Contract Number | String |  |
| 14 | Corporate Author | String |  |
| 15 | Date | date |  |
| 16 | Descriptors | String |  |
| 17 | Last Update | String |  |
| 18 | Loan Date | date |  |
| 19 | Loan Due Date | Date |  |
| 20 | Maps | String |  |
| 21 | Notes | String |  |
| 22 | PaperOrSpeech | String |  |
| 23 | PreparedFor | String |  |
| 24 | Publisher | String |  |
| 25 | RecordNumber | String |  |
| 26 | ReferenceNumber | String |  |
| 27 | Series | String |  |
| 28 | Aisle | String |  |
| 29 | Bay | String |  |
| 30 | Shelf | String |  |
| 31 | Shelf Barcode | String |  |
| 32 | Site | String |  |
| 33 | Title | String |  |
| 34 | Type | String |  |
| 35 | Last Name | String |  |
| 36 | First Name | String |  |
| 37 | Record Series Code | String |  |
| 38 | Site Name | String |  |
| 39 | Site Number | String |  |

### setRecordRequests

This call creates/edits/deletes/updates record request records. The directory structure is the following. All record requests are stored under a directory node called “Record Requests”. This node and the dates are storage record types. Request Date is parsed in year, month and day. Then the directory node is an electronic record of type Record Request. The name of the directory node is a combination of the Request Company and up to 40 chars of the Info Requested field.

Record Requests

YYYY

MM

DD

Request Company Info Requested (up to 40 chars)

**POST:**

{webserver}/formsservice/setRecordRequests/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | string |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Request ID Number | string |  |
| 10 | Request Date | date |  |
| 11 | Due Date | date |  |
| 12 | Finished Date | date |  |
| 13 | Request Company | String |  |
| 14 | Request Last Name | String |  |
| 15 | Request First Name | String |  |
| 16 | Address | String |  |
| 17 | City | String |  |
| 18 | State | String |  |
| 19 | Zipcode | String |  |
| 20 | Email | String |  |
| 21 | Received By | String |  |
| 22 | Assigned To | String |  |
| 23 | Info Requested | String |  |
| 24 | Notes | String |  |

### setSignatures

**GET:**

{webserver}/formservice/setSignatures/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D”=this is a invoice detail item |
| 3 | objectId | string | This is the timecard detail objectId. When you do an update you must have a header. |
| 4 | objectType | string | This is the timecard detail objectType |
| 5 | MobileRecordId | string | This is the group responsible for the record |
| 6 | FunctionalGroupName | String | This is used by the local device database and is generated by deviceid+timestamp |
| 7 | organizationName | string | This is the organization name to whom the customer belongs |
| 8 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 9 | Document Title | string |  |
| 10 | Issuing Authority | string |  |
| 11 | Expiration Date | Date |  |
| 12 | Signature Date | Date |  |
| 13 | Signature Name | String |  |
| 14 | Description | string |  |
| 15 | ItemType | string |  |
| 16 | Document Type | String |  |
| 17 | Document Date | Date |  |
| 18 | Reviewed By | String |  |
| 19 | Reviewed Date | Date |  |
| 20 | parentObjectId | String |  |
| 21 |  |  |  |

## MapService

The map services provides several geocoding and distance ordering algorithms. For the most part it is better to calculate things on the server and let the mobile devices store the results.

### getZipcodePlus4

Given an address you want to call the USPS web service and get the 9 digit zipcode for an address. This is sometimes called “zip + 4”.

**GET:**

{webserver}/mapservice/getZipcodePlus4/{login}/{password}/{address}/{city}/{state}

**Returns:**

E+ error Message

9digits=zip+4

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Address | Address of the location |
| City | city of the location |
| State | State of the location (2 chars) |

### getGpsCoordinatesFromAddress

Given an address you want to call google geocoding to get the the gps coordinates.

**GET:**

{webserver}/mapservice/getGpsFromAddress/{login}/{password}/{address}/{city}/{state}

**Returns:**

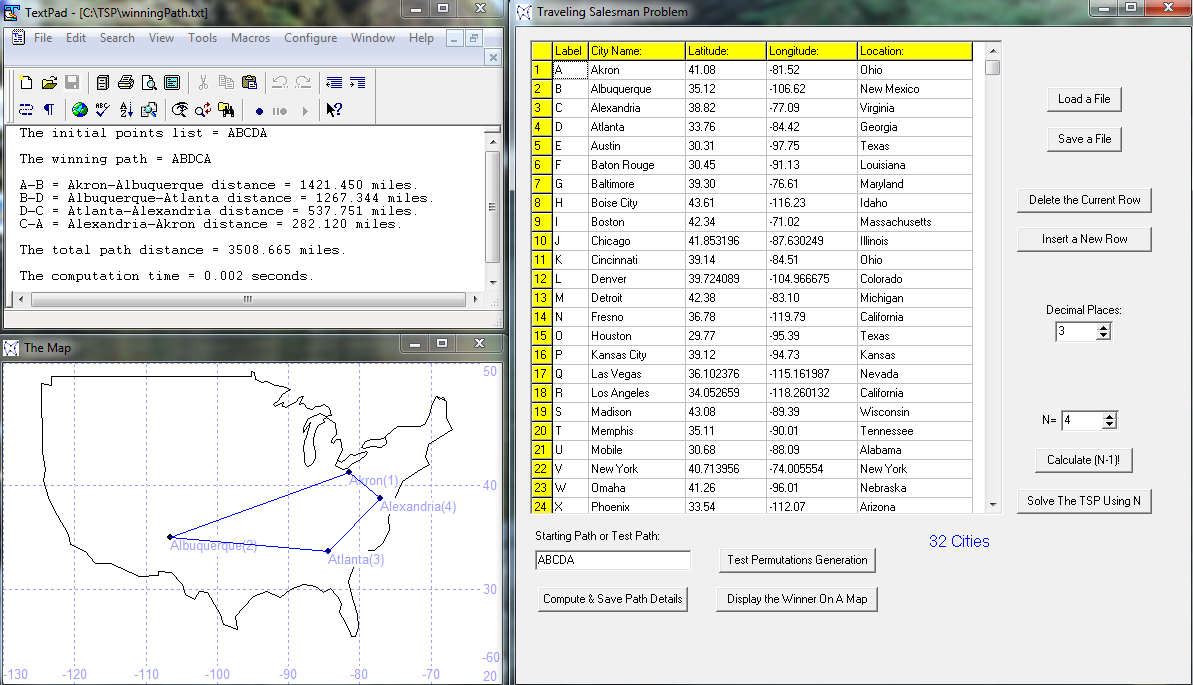
Latitude, Longitude

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Address | Address of the location |
| City | city of the location |
| State | State of the location (2 chars) |

### getOptimalRoute

Given a csv file containing a list or gps coordintates find shortest route. In the figure below the first point is the starting point and the ending point. We will modify City Name to Description. The location is not needed in the call. Each label is a single letter and must be unique. For the return you will get the winning path string which is a string of letters that represents the optimal route. We are optimizing for distance.



**GET:**

{webserver}/mapservice/getOrderedList/{login}/{password}/

CSV file

“Letter”, “Description”,”Latitude”,”Longitude”

**Returns:**

String of letters

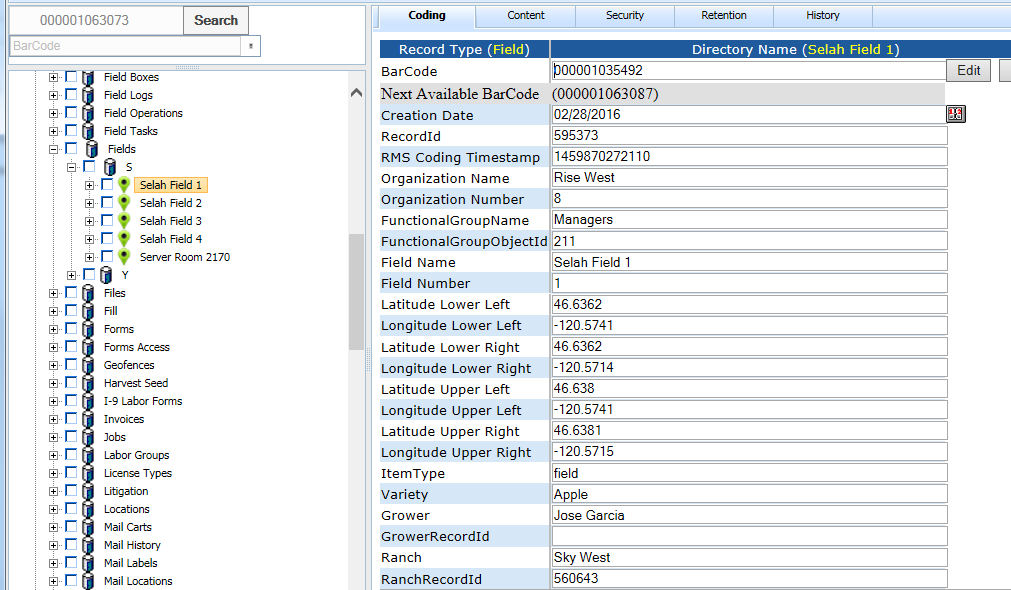
Number+error message string

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setFields

This call creates/updated fields that are use in tracking the time and location of labor groups. The records are organized in the directory under a node called fields and then grouped by a storage container using the first later of the ranch name as shown in the following example. The directory node name is formed by the ranch name field number.



**GET:**

{webserver}/mapservice/setFields/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | String | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | String | “H” - Indicates this is a invoice header item |
| 3 | objectId | String | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | String | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | String | This is the group responsible for the record |
| 7 | organizationName | String | This is the organization name |
| 8 | organizationNumber | String | This is the organization number |
| 9 | Grower | string |  |
| 10 | GrowerRecordid | string |  |
| 11 | Ranch | string |  |
| 12 | RanchRecordId | string |  |
| 13 | Field Name | String | This is a name that has to be unique in organization for lookup |
| 14 | FieldRecordId | String |  |
| 15 | Location | String |  |
| 16 | Latitude | String |  |
| 17 | Longitude | String |  |
| 18 | ItemType | String |  |
| 19 | Status | String |  |
| 20 | Estimated Yield | String |  |
| 21 | Harvest Yield | String |  |
| 22 | Acres | String |  |
| 23 | Beds | string |  |
| 24 | Product Category | String |  |
| 25 | Product Type | string |  |
| 26 | Beds Harvested | Number |  |
| 27 | Beds Remaining | Number |  |
| 28 | Beds Total | Number |  |

**Returns:**

Number of locations created/modified/deleted

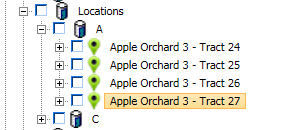
Number+error message string

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setLocations

This call creates locations that are use in tracking the time and location of labor groups. The records are organized in the directory under a node called locations and then grouped by a storage container using the first later of the location name as shown in the following example.



**GET:**

{webserver}/mapservice/setLocations/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | organizationName | string | This is the organization name |
| 8 | organizationNumber | string | This is the organization number |
| 9 | Location | string | This is the location where workers will go |
| 10 | Latitude | string | Latitude of new location |
| 11 | Longitude | String | Longitude of new location |

**Returns:**

Number of locations created/modified/deleted

Number+error message string

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setRanches

This call creates/update ranches that are used in tracking all field activities. The records are organized in the directory under a node called Ranches and then grouped by a storage container using the first letter of the ranch name as shown in the following example. The directory node name is formed by the ranch coding field.



**GET:**

{webserver}/mapservice/setRanches/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | String | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | String | “H” - Indicates this is a invoice header item |
| 3 | objectId | String | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | String | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | String | This is the group responsible for the record |
| 7 | organizationName | String | This is the organization name |
| 8 | organizationNumber | String | This is the organization number |
| 9 | Ranch | string |  |
| 10 | Grower | string |  |
| 11 | GrowerRecordId | string |  |
| 12 | Location | String |  |
| 13 | Latitude | String |  |
| 14 | Longitude | String |  |
| 15 | ItemType | String |  |
| 16 | Address | String |  |
| 17 | City | String |  |
| 18 | State | String |  |
| 19 | Zipcode | string |  |
| 20 | Country | string |  |

**Returns:**

Ranch Name created or

Number+error message string

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

## Mobile Service

The mobile services handle requests from the Motorola MC9090 barcode/rfid scanner.

### getSensorData

This returns object ids of sensors for a given location and sensor type.

**GET:**

{webserver}/mobileservice/getSensorData/{login}/{password}/{LocationName}/{SensorType}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique string to identify user |
| password | User’s password |
| Location | This is the location name |
| SensorType | Moisture, wind, rain |

### uploadFolderFile

Upload a file to a web storage file specified by the category path.

**POST:**

{webserver}/mobileservice/uploadFolderFile/{login}/{password}/{path}

**EXAMPLE:**

curl -k -X POST -F media=@scan.txt {webserver}/mobileservice/uploadFolderFile/login/password/path

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Path | Category1….category5 |

### uploadScanData

This restful interface is used to collect data from the mobile device and then pass this information on to the rms for processing. The data on the device is formatted as a single string with a space character used to separate the commands/data (tokens). The web service replaces the space character with a carriage return and line feed and calls scanBean.scanProcess(scanData) located in com.rco.jsp.CScanBean.

**POST:**

{webserver}/mobileservice/uploadScanData/{login}/{password}/{scanData}/

**EXAMPLE:**

{webserver}/mobileservice/uploadScanData/login/password//RCVPRT NONE 1 45-602-1-2-3 500-0001-000 ENDSCN/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| scanData | This is a string of commands/data separated by a space. |

**Steps**

Create scanauditrecord and keep objectId and objectType which will be used in setRecordCoding.

Get user First Name, Last Name and userid from Login in the CBillingOperations module – sessionLogin(login, password)

Get 1st operation from DataFile (trim preceding spaces and get 1st token)

Date = get Current Date

Time = get current time

Description = copy up to 2k chars of DataFile

If operation = RCVPRT

1. Scan start operation RCVPRT

2. Scan operation RMA or PO

3. Scan item Label = item number + record id

4. Scan Quantity

5. Scan New Location

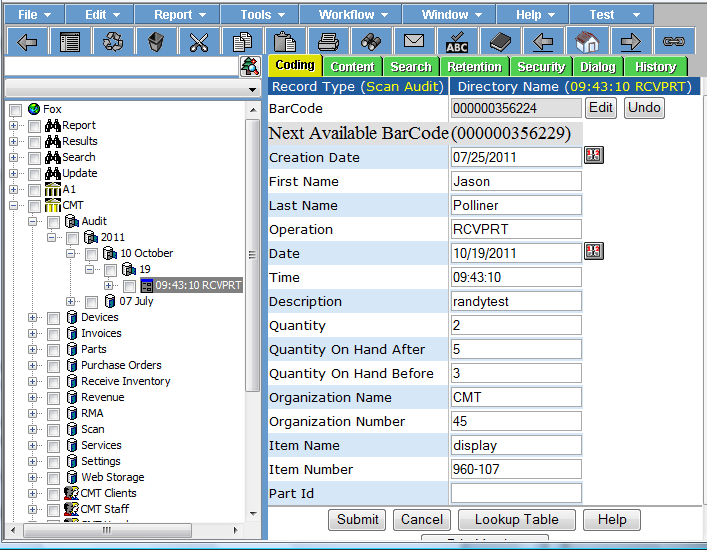
6. Scan end operation ENDSCN

When you have a receive part and you have an rma then you can find the location of the part from the record id in the item label. You look up whether the part is tracked or not from the library using the item number. You can tell whether the rma is from the customer or vendor by checking the deployment of the record id.

Find the Quantity on Hand for the given item number from the library and save into Quanity on Hand Before.

In library Quantity on Hand After = Quantity on Hand before + Quantity

Set all the coding fields with data collected.



### uploadScanFile

This restful interface is used to collect data from the mobile device and then pass this information on to the rms for processing. The web service creates an array of strings and calls scanBean.scanProcess(scanData) located in com.rco.jsp.CScanBean.

**POST:**

{webserver}/mobileservice/uploadScanFile/{login}/{password}

**EXAMPLE: curl post command to upload a scan file:**

curl -k -X POST -F media=@scan.txt {webserver}/mobileservice/uploadScanFile/login/password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Scan.txt | This could be any file with any name that contains the scan data information to be uploaded to the server. An example of the file content is as following:  NEWKIT P1-STEEL 1 45-902-1-2-3 ENDSCN  Note that each field is separated by a space |

### uploadScanLog

This call was designed to capture all scanning in the shipping and receiving from the mobile device. This will eliminate the need for the scan history. On the mobile this file is saved locally and the mobile keeps sending this to the server until it gets the file transferred correctly with the same number of lines it sends up. The server web services will count the number of lines in the csv file and return this or an error. On the server the log will be in a folder called uploadscanlog. Under this is a folder org number org name. Then you have Ship or Receive.

The mobile will send a file name tabletNumber-date-time.txt

Where date = YYYYMMDD and Time = HHMMSS

On the server the information is saved in a folder called UploadScanLog.

**POST:**

{webserver}/mobileservice/uploadScanLog/{login}/{password}/{NumberLinesInCsvFile}/

**EXAMPLE: curl post command to upload a scan file:**

curl -k -X POST -F media=@scan.txt {webserver}/mobileservice/uploadScanLog/login/password/{NumberLinesInCsvFile}/{Command}

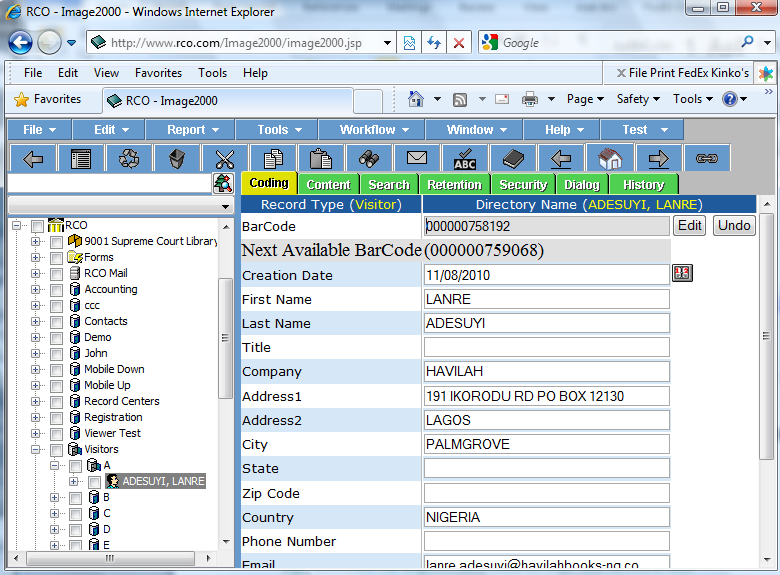
**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| NumberLinesInCsvFile | This is the number of lines in the csv file and it better match exactly |
| Command | this is shipping or receiving |

|  |  |
| --- | --- |
| Scan.txt | This could be any file with any name that contains the scan data information to be uploaded to the server. An example of the file content is as following:  Tablet, buildNumber, Date, time, fleet, docNumber, status, itemNumber, description, Serial Number  docNumber = crma# or invoice# |

### createVisitor

At a show a sales person scans a badge with visitor information encoded with a PDF417 barcode. The mobile scan device saves the badge scans and then uploads the information to the rms server. This restful interface is used to create a single visitor in folder called Visitors for the given organizationNumber (see following image). We separate visitors by the first letter of last name. The visitor is a visitor record type and all the fields from the badge are set in the visitor coding fields.



**POST:**

{webserver}/mobileservice/createVisitor/{login}/{password}/{visitor}/

**EXAMPLE:**

{webserver}/mobileservice/createVisitor/login/password/visitor

**ARGUMENTS:**

firstName can be an empty string

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| visitor | This is a single visitor data structure |

### getDeviceId

This function returns a coding field called Device Id for a given login. This function is used in creating a mobile invoice id number to uniquely identify invoices when in the field and no carrier or internet connection is available.

**GET:**

{webserver}/mobileervice/getDeviceId/{login}/{password}/

**Returns**

DeviceId

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setSalesTypes

This function creates or edits one or more sales types.

**POST:**

{webserver}/ mobileservice /setSalesTypes/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | LicenseCode | string | Obtained from government entity |
| 8 | LicenseType | string | Obtained from government entity |
| 9 | LicenseDescription | string | Obtained from government entity |

### setSalesRecords

This function creates, edits a license header and one or more details. A business can have a licenses with one or more detail that describes a different power of the license. As an example a business can have beer and wine license and a distilled spirits license.

**POST:**

{webserver}/ mobileservice /setSalesRecords/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | licenseNumber | string | Obtained from government entity |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. | |
| 2 | Flag | string | “D” =this is an invoice detail item |
| 3 | objectId | string | This is the objectId for the detail if you are updating a record otherwise it is blank. |
| 4 | objectType | string | This is the objectType for the detail if you are updating a record otherwise it is blank |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp | |
| 6 | FunctionalGroupName | string | This is the group responsible for the record | |
| 7 | licenseNumber | string | Set by government entity |
| 8 | Address1 | string | Location of store |
| 9 | Address2 | string | More address information |
| 10 | City | String | City |
| 11 | State | string | State where store is located |
| 12 | Zipcode | string | Zipcode of store |
| 13 | Latitude | string | Gps location |
| 14 | Longitude | String | Gps location |
| 15 | Licensecode | String | Government issued |

Each data line is separated by a CRLF. Data fields within a line are separated by a comma. All data elements are surrounded by quotes. You can have a header record and no details if you are doing an update on the header. To update a detail you must have the header record.

If the Invoice Numbrer or Mobile Invoice Number exist then you are doing an update operation otherwise you are doing a record creation.

Example of Curl command:

curl -k -X POST -F [media=@fields.txt](mailto:media=@fields.txt) [{webserver}/](https://www.rcofox.com/Image2000/rest/)quickbookservice/setInvoices/login/password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

## Part Service

This section deals with parts. Quickbooks needs to add new parts to a store and get a list of the parts in a store.

### changeItemNumber

This call changes the item number and is used for auditing the move operation which is a processing operation like packing. The calle has to pass a combination of some optional paramaters.

**POST:**

{webserver}/partservice/changeItemNumber/{login}/{password}/{objectId}/{objectType}/{recordId}/{scanCode}/{OldItemNumber}/{NewItemNumber}/{description}/{datetime}/{LocationRecordId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| objectId | Opt |
| objectType | Opt |
| recordId | Opt |
| ScanCode | Opt this is the manufacturer serial number barcode you scan |
| OldItemNumber | This is the produce pack item number that you scanned |
| OldItemDescription | That was the old item number |
| NewItemNumber | Base on the accounting item number so if you have an empty product case and you scan strawberries into the product pack now the item is a strawberry pack but the record type is still a product pack |
| NewItemDesciption | This is the new item number of finished product |
| ChangeReason | This is the reason for the change like packing |
| Datetime | Ansi format YYYY-MM-DD HH:MM:SS.SSS |
| LocationRecordId | Location where the item is at the time you changed the item number |

### createNonTrackedPart

This call creates a single non-tracked item in the library for a given login organizationNumber. On the server

**POST:**

{webserver}/partservice/createNonTrackedPart/{login}/{password}/{itemNumber}/{description}/{deployment}/{VendorName}/{ManufacturerSerialNumber}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| itemNumber | This is item number in the library |
| Description | This describes the item |
| VendorName | Vendor name |
| ManufacturerSerialNumber | This is the serial number for a tracked part created by the manufacturer to uniquely identify each of their parts. |

### createPart

This restful interface is used to create a single part in Inventory under the Parts storage container for a give organizationNumber. Within the Parts container there is an inventory storage container where the parts are stored. We separate parts by the first letter.

**POST:**

{webserver}/partservice/createPart/{login}/{password}/{itemNumber}/{quantity}/{LocationRecordId}/{deployment}/{VendorName}/{ManufacturerSerialNumber}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| itemNumber | This is item number in the library |
| Quantity | Number of items to create |
| locationRecordId | Where the item will go |
| Deployment | General state of where part is located |
| VendorName | Vendor name |
| ManufacturerSerialNumber | This is the serial number for a tracked part created by the manufacturer to uniquely identify each of their parts. |
| PartType | This is either (part, adjustment or phantom) with the default being part if argument is missing. |

### createParts

This function takes a csv file and creates part in Inventory under the Parts storage container for a give organizationNumber. Within the Parts container there is an inventory storage container where the parts are stored.

**POST:**

{webserver}/partservice/createParts/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. | |
| 2 | Flag | string | “H” =this is an part header item |
| 3 | objectId | string | This is the objectId for the detail if you are updating a record otherwise it is blank. |
| 4 | objectType | string | This is the objectType for the detail if you are updating a record otherwise it is blank |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp | |
| 6 | FunctionalGroupName | string | This is the group responsible for the record | |
| 7 | Organization Name | String |  | |
| 8 | Organization Number | String |  | |
| 9 | itemNumber | String |  | |
| 10 | quantity | String |  | |
| 11 | locationRecordId | String |  | |
| 12 | Deployment | String |  | |
| 13 | vendorName | String |  | |
| 14 | manufacturerSerialNumber | String |  | |
| 15 | objectTypeOptional | string |  | |

### createPart

This restful interface is used to create a single part in Inventory under the Parts storage container for a give organizationNumber. Within the Parts container there is an inventory storage container where the parts are stored. We separate parts by the first letter.

**POST:**

{webserver}/partservice/createPart/{login}/{password}/{itemNumber}/{quantity}/{LocationRecordId}/{deployment}/{VendorName}/{ManufacturerSerialNumber}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| itemNumber | This is item number in the library |
| Quantity | Number of items to create |
| locationRecordId | Where the item will go |
| Deployment | General state of where part is located |
| VendorName | Vendor name |
| ManufacturerSerialNumber | This is the serial number for a tracked part created by the manufacturer to uniquely identify each of their parts. |
| PartType | This is either (part, adjustment or phantom) with the default being part if argument is missing. |

### createPartWithCoding

This restful interface is used to create a single part in Inventory under the Parts storage container for a give organizationNumber. Within the Parts container there is an inventory storage container where the parts are stored. We separate parts by the first letter. This is the same as createPart except that you have 2 extra arguments PTN and Activation Date.

**POST:**

{webserver}/partservice/createPart/{login}/{password}/{itemNumber}/{quantity}/{LocationRecordId}/{deployment}/{VendorName}/{ManufacturerSerialNumber}/{PTN}/{ActivationDate}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| itemNumber | This is item number in the library |
| Quantity | Number of items to create |
| locationRecordId | Where the item will go |
| Deployment | General state of where part is located |
| VendorName | Vendor name |
| ManufacturerSerialNumber | This is the serial number for a tracked part created by the manufacturer to uniquely identify each of their parts. |
| PTN | This is the phone number |
| Activation Date | When the device was activated |

### createTrackedPart

### createStore

This restful interface creates a ship store header used to send items to the customer or vendor.

**POST:**

{webserver}/partservice/createStore/{loginId}/{password}/{locationRecordId}/{itemType}/{storeId}/{storeName}/{storeNumber}/{invoiceNumber}/{vendorRmaNumber}/{destinationStoreNumber}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| locationRecordId | This is the carrier location recordId |
| itemType | this should be store |
| storeId | this is a further unique store identification Ex hac# |
| storeName | This is the carrier |
| storeNumber | This is the trackingId |
| invoiceNumber | If you have an invoice then items are going to customer |
| vendorRmaNumber | If you have a vendor rma then items are going to vendor |
| desitinationStoreNumber | If invoiceNumber is not blank then lookup store name from the invoice and then look up store number (getRecordIdsAll)  If vendorRma is not blank then store number from getVendorHeaderId |

### findTruck

Give the license number return the truck gps latitude and longitude.

**GET:**

{webserver}/partservice/findTruck/{login}/{LicenseNumber}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Logon | Unique string to identify user |
| password | User’s password |
| LicenseNumber | This is a unique vehichle license number |

**Returns**

objectId, objectType, latitude, longitude

### getAllItemLocationsByStore

This call will return all the locations for a given store. The return includes objectId, objectType, recordId.

**GET:**

{webserver}/partservice/getStoreShelfLocations/{login}/{password}/{storeNumber}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| storeNumber | This is the store number for the given login |

### getCycleCount

This call will return the cycle count for a given user that the mobile device can use to obtain the counts.

**GET:**

{webserver}/partservice/getCycleCount/{login}/{password}/{userRecordId}/{date}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| UserRecordId | The cycle counter’s user record id |
| Date | Normally this is todays date and if blank then all cycle counts for this user starting from 0 |

### getInventoryCounts

This call will return a list of counts for all items if ItemNumber is blank.

**GET:**

{webserver}/partservice/getInventoryCounts/{login}/{password}/{locationRecordId}{ItemNumber}/{MaxItemsReturned}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| locationRecordId | This is where the item is located |
| ItemNumber | this is the library item |
| MaxItemsReturned | If ItemNumber is blank and you specify 500 then it will return only 500 quantities. If blank then it returns all item numbers and their quanties. |

### getItemIdInfo

This restful interface is used to get object id, object type for a given item id and location.

**GET:**

{webserver}/partservice/getItemIdInfo/{login}/{password}/{recordId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Logon | Unique string to identify user |
| password | User’s password |
| recordId | Unique rms system recordId for the given item |

### getPacking

This call returns all the invoice header and details for a given truck and given date range.

**GET:**

{webserver}/partservice/getPacking/{login}/{VehicleLicenseNumber}/{fromDate}/{toDate}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Logon | Unique string to identify user |
| password | User’s password |
| VehicleLicenseNumber | This is the truck vehicle license number |
| fromDate | The from and to date can be equal to get a specific date |
| toDate | Date format is YYYYMMDD |

### getPartIdInfo

This restful interface is used to get object id, object type for a given part id.

**GET:**

{webserver}/partservice/getPartInfo/{login}/{password}/{partId}/

**ARGUMENTS:**

No empty arguments allowed.

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| partId | This is a unique rms id assigned to an item number that has tracking=true and might also have kit=true |
| objectTypePart | this can be null ask Randy what this is used for |

### getPendingServiceTickets

Get all the pending service tickets where departure time is blank for the given logon’s organization.

**GET:**

{webserver}/partservice/getPartInfo/{login}/{password}/

**ARGUMENTS:**

No empty arguments allowed.

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |

### getStoreHeaderId

This is used to get the store header id for a given storeNumber and organization.

**GET:**

{webserver}/partservice/getStoreHeaderId/{login}/{password}/{organizationumber}/{storeNumber}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| organinzationNumber | This is the user’s organization number |
| storeNumber | this is the store number where you want to get the objectId and objectType |

### getStoreNumberForLocationRecordId

This call all the coding fields for a given RecordId.

**GET:**

{webserver}/partservice/getStoreNumberForLocationRecordId/{login}/{password}/{LocationRecordId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| LocationRecordId | This is the recordId from where all the coding will be returned |

### getStoreInventoryList

This restful interface is used to get the part count of each part type of a given store number. This is returning an array of each different part with the count for each part.

**GET:**

{webserver}/partservice/getStoreInventoryList/{login}/{password}/{storeNumber}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| storeNumber | This is the number of the store you want to get the parts inventory list. |

### getStorePartCount

This restful interface is used to get the part count of each part type of a given store number.

**GET:**

{webserver}/partservice/getStorePartCount/{login}/{password}/{storeNumber}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| storeNumber | This is the number of the store you want to count the parts. |

### getStorePartIdsAll

This restful interface is used to get all the part ids for a given store number.

**GET:**

{webserver}/partservice/getStorePartIdsAll/{login}/{password}/{storeNumber}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| storeNumber | This is the given store number from which you want all the part ids. |

### getStorePartIdsUpdated

This restful interface is used to get all the part ids for a given stores that are greater than the given coding timestamp.

**GET:**

{webserver}/partservice/getStorePartIdsUpdated/{login}/{password}/{storeNumber}/{codingTimestamp}

Use 0 for the first time to give all part ids but if we would supply a timestamp then the system would give us only those parts that have a greater timestamp.

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| storeNumber | This is the given store number from which you want all the part ids. |

### getTruckPacking

This call returns all the packing header and details for a given truck for the last packing.

**GET:**

{webserver}/partservice/gettruckPacking/{login}/{VehicleLicenseNumber}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Logon | Unique string to identify user |
| password | User’s password |
| VehicleLicenseNumber | This is the truck vehicle license number |

### movePart

This call will move a tracked or non-tracked part from the old location to the new location. When moving a non-tracked part the system will decrement the quantity on hand at the old location and increment the quantity on hand at the new location. If the non tracked part does not exist at the new location one will be created.

**POST:**

{webserver}/partservice/movePart/{login}/{password}/OldRecordIdNontracked}/{newLocationRecordId}/{partRecordId}/{quantity}/{deployment}/{ScanCode}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| OldocationRecordIdNonTracked | This is the record id of the location where you are moving the non tracked part from. Tracked parts do not need to fill in this field and it is better to leave this field blank |
| newLocationRecordId | This is the record id of where you are moving the part |
| partRecordId | This is the record id of the part you are moving |
| Quantity | Number of parts to move. A tracked part must have a quantity of 1 |
| Deployment | This is a coding field that helps distinguish the status of a part. Is the part in the factory, depot, customer, vendor |
| ScanCode | If newLocationRecordId is unknown then ScanCode which is derived from Item Primary Key Coding Field in the Database record type under org/settings Example: recordId or manufacturer serial number. Please note that if you have the recordId then you have put that in newLocationRecordId and ScanCode |

// lot of fun validation & error handling

If substring(part or item id,2) equals “@@” then

// this is an tracked part case

Move(oldLocation,NewLocation)

Else // this is a non-tracked part

{

If storage bin substring(item number,1) does not exist create it

If !part.exist create part with quantity on hand = 0

decrement part.quantity on hand by quantity at old location

Increment part.quantity on hand by quantity at new location

}

### moveParts

Move a list of parts given a csv file (see next table) to a new location. The moveParts call will return an **array** of "NodeInfoBase" objects, each element represents one of the parts moved.

public class NodeInfoBase  
{  
 // data members  
 protected int treeId = -1;  
 protected int objectId = -1;  
 protected String objectType = "";  
 protected String name = "";  
 protected int parentTreeId = -1;  
 protected boolean isEncrypted = false;  
 protected int encryptMode = -1;  
 protected String scanCode = ""; // new member  
}

**POST:**

{webserver}/partservice/moveParts/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | OldLocationRecordIdNonTracked | string | This is the record id of the location where you are moving the non tracked part from. Tracked parts do not need to fill in this field and it is better to leave this field blank |
| 2 | newLocationRecordId | string | This is the record id of where you are moving the part |
| 3 | partRecordId | string | This is the record id of the part you are moving |
| 4 | Quantity | string | Number of parts to move. A tracked part must have a quantity of 1 |
| 5 | Deployment | string | This is a coding field that helps distinguish the status of a part. Is the part in the factory, depot, customer, vendor |
| 6 | ScanCode | string | If newLocationRecordId is unknown then ScanCode which is derived from Item Primary Key Coding Field in the Database record type under org/settings Example: recordId or manufacturer serial number. Please note that if you have the recordId then you have put that in newLocationRecordId and ScanCode |

### movePartsOffline

Move a list of parts given a csv file (see next table) to a new location. The call will return an array of the scan codes received.The rms processes all offline moves starting at 11pm each day.

**POST:**

{webserver}/partservice/movePartsOffline/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

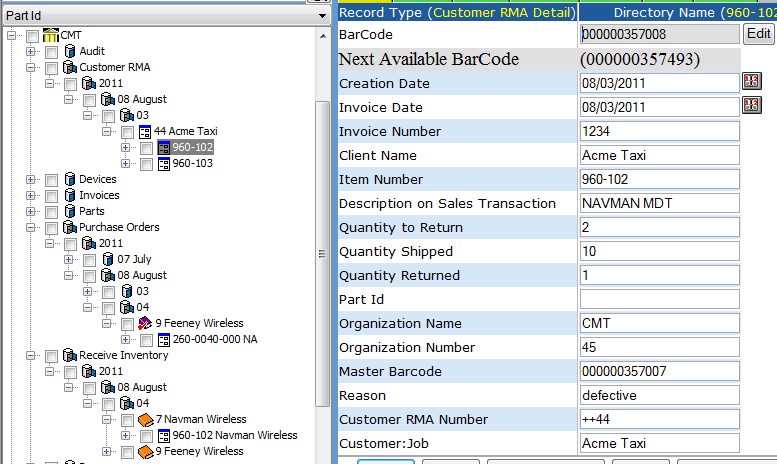
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | OldLocationRecordIdNonTracked | string | This is the record id of the location where you are moving the non tracked part from. Tracked parts do not need to fill in this field and it is better to leave this field blank |
| 2 | destinationRecordId | string | This is the parent in the tree. The container |
| 3 | partRecordId | string | This is the record id of the part you are moving |
| 4 | Quantity | string | Number of parts to move. A tracked part must have a quantity of 1 |
| 5 | Deployment | string | This is a coding field that helps distinguish the status of a part. Is the part in the factory, depot, customer, vendor |
| 6 | ScanCode | string | If newLocationRecordId is unknown then ScanCode which is derived from Item Primary Key Coding Field in the Database record type under org/settings Example: recordId or manufacturer serial number. Please note that if you have the recordId then you have put that in newLocationRecordId and ScanCode |
| 7 | Timestamp | string | This is the timestamp when the scan was mded |

### receivePartHeader

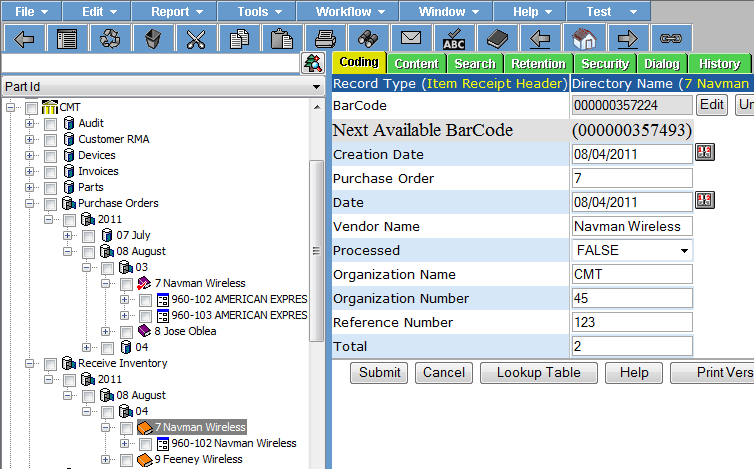
This restful interface is used to receive a part in a location. There are 2 types of parts coming into the factory. There are several different types of parts coming into the factory. For each receive purchase order item you update the library item number quantity on hand.

|  |  |
| --- | --- |
| **Part** | **Description** |
| New part | A new part comes from a vendor defined in the accounting system and a purchase order number exists that defines the quantity, item number. The vendor has to put the title purchase order followed on the shipment box. The mobile unit sets originType=PURCHASE |
| Customer rma | The factory generates a unique ++number from a customer rma forms. The form consists of a header and detail form. The mobile unit sets originType=CUSTOMERRMA |
| Vendor rma | When the factory ships parts to the vendor and the vendor returns the part they must have the rma number on the box. The mobile unit sets originType=VENDORRMA |

If you have an rma then you have to either increment the quantity at old location and increment the quantity on the new location for a non-tracked part or you move the tracked part from the old location to the new location. You have to increment the quantity returned coding field in the rma form.



If origin type is Purchase Order then check if the itemReceiptHeader exists for the given and origin number. If the header does not exist then create the header as shown in the following figure and create the itemReceiptDetail.



**GET:**

{webserver}/partservice/receivePartHeader/{login}/{password}/{origin number}/{origin type}/{receiptDate}/{referenceNumber}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| origin number | This is the 1st number scanned |
| origin type | This is a combo box that describe what the receive number is which can be {factory rma, vendor rma or purchase order}. The origin type is blank but if we detect a ++ we will automatically fill in factory rma. |
| receiptDate | This is when you received the items |

// lot of validation and error handling left as an exercise for the reader.

If (originType == CUSTOMERRMA) then

Lookup up the customer rma number header by

customerRmaNumber=originNumber & return record id of customerRmaHeader

Move(oldLocation,NewLocation)

If (originType == VENDORRRMA) then

Lookup up the vendor rma number header by

vendorRmaNumber=originNumber & return record id of vendorRmaHeader

Move(oldLocation,NewLocation)

Else If (originType == PURCHASE) then

{

lookupPurchaseOrderHeader by purchaserOrderNumber=originNumber

get the vendor name from purchaseOrderHeaderRecord

create the itemReceiptHeader and return the record id of the itemReceiptHeader

plug in the vendorName

}

Else If (originType == VENDORRMA) then

Lookup rma.header.invoiceNumber.total

### receivePartDetail

This restful interface is used to receive a part in a location. There are 2 types of parts coming into the factory. There are several different types of parts coming into the factory. For each receive purchase order item you update the library item number quantity on hand.

It is a weird format:

The first item in the returned array is the node-info for the "Item Receipt Detail" record.

The last item in the returned array is the node-info for the "Purchase Order Detail" record.

The returned array items in between the first and last are the node-info for the new Parts received.

**GET:**

{webserver}/partservice/receivePartDetail/{login}/{password}/{objectIdHeader}/{objectTypeHeader}/{origin number}/{origin type}/{receiptDate}/{customer}/{itemNumber or itemLabel\*\*}/{ManufacturerSerialNumber}/{quantity}/{LocationRecordId}/

\*\* - itemNumber when you have a Purchase Order or recordId if RMA

**returns:**

array of nodeinfobase which can be used to get itemLabels to generate the part labels

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| objectIdHeader | This is returned from the receivePartHeader call |
| objectTypeHeader | This is returned from the receivePartHeader call |
| origin number | This is the 1st number scanned |
| origin type | This is a combo box that describe what the receive number is which can be {factory rma, vendor rma or purchase order}. The origin type is blank but if we detect a ++ we will automatically fill in factory rma. |
| receiptDate | This is the you received the part date |
| Customer | This is if you are specifying a designated customer for a job |
| \*\*ItemNumber or itemLabel | ItemNumber if you have purchase order or itemLabel if you have an customer or vendor RMA |
| ManufacturerSerialNumber | This is the serial number for a tracked part created by the manufacturer to uniquely identify each of their parts. |
| Quantity | Number of the item number received |
| LocationRecordId | This is where the item is getting stored |

// lot of validation and error handling left as an exercise for the reader.

If (originType == CUSTOMERRMA) then

Move(oldLocation,NewLocation)

Have handle to header record

getBarcodeOfHeaderRecord

lookupDetailRecord by MasterBarcode & partOrItemId

update Item Return Count

If (originType == VENDORMA) then

Move(oldLocation,NewLocation)

Have handle to header record

getBarcodeOfHeaderRecord

lookupDetailRecord by MasterBarcode & partOrItemId

update Item Return Count

Else If (originType == PURCHASE) then

createPart(NewLocation)

createItemReceiptDetail for the part

Else If (originType == VENDORRMA) then

Lookup rma.header.invoiceNumber.total

### PartsImport

This reads a csv file created on the workstation that contains a list of part record ids and the part key (normally the manufacturer serial number). The system will

**POST:**

{webserver}/partservice/PartsImport/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | PartRecordId | string | This is the recordId of the part |
| 2 | Key | string | This is the coding field used as the key. Ex S/N |

### recodeTrackedItem

When an item is incorrectly coded the operator wants to fix the item on the fly. This call does several things.

1. Change the number of the directory node to New Item Number Description
2. Change the item number to New Item Number
3. Change the description, description on Sales Transaction, description of Purchase transaction
4. In the library decrement the old item number quantity on hand
5. In the library increment the inew item number quantity on hand

**POST:**

{webserver}/partservice/recodeTrackedItem/{login}/{password}/{ObjectId}/{ObjectType}/{OldItemNumber}/{NewItemNumber}

### setAdjustments

This function takes a csv file containing a table of rows where the columns are:

Item Number,Item Description,Qty,Store Name,Store Number,New,CRMA. You have to add New & CRMA and then create an adjustment part in the store if it does not exist subtracting the difference from the quantity on hand. You also have to adjust the library quantity on hand for the given part.

**GET:**

{webserver}/partservice/setErrorMessage/{login}/{password}/{InventoryDate}/{ReportDate}/

Returns:

Success or Failure

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| InventoryDate | ??? |
| ReportDate | ??? |

### setChangeItemNumbers

This call changes the item numbers of multiple records and is used for auditing the move operation which is a processing operation like packing. The call pass a csv file.

**POST:**

{webserver}/partservice/setChangeItemNumbers/{login}/{password}/

**CSV File:**

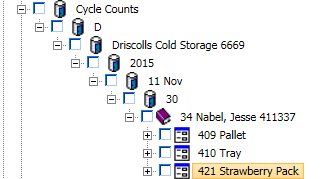
**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | RecordId | String |  |
| 9 | ScanCode | String | manufacturer serial number barcode you scan |
| 10 | OldItemNumber | String | This is the empty packaging |
| 11 | OldItemDescription | String | This is the description of the empty packaging |
| 12 | NewItemNumber | String | Based on the accounting item number so if you have an empty product case and you scan strawberries into the product pack now the item is a strawberry pack but the record type is still a product pack |
| 13 | NewItemDesciption | String | This is the retail package description |
| 14 | ChangeReason | String | This is the reason for the change like packing |
| 15 | Datetime | String | Ansi format YYYY-MM-DD HH:MM:SS.SSS |
| 16 | LocationRecordId | String | Location where the item is at the time you changed the item number |

### setCows

### setCycleCounts

This call is used to create and update cycle count worksheets. The directory struction is shown in the following figure where you have the first letter of the store and then year, month, day followed by the cycle count header and details for each user.



**GET:**

{webserver}/partservice/setCycleCounts/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Organization Name | String |  |
| 8 | Organization Number | string |  |
| 9 | UserRecordId | String |  |
| 10 | First Name | string |  |
| 11 | Last Name | string |  |
| 12 | Date | Date |  |
| 13 | Store Name | String |  |
| 14 | Store Number | String |  |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a detail item |
| 3 | objectId | string |  |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | UserRecordId | String |  |
| 10 | First Name | string |  |
| 11 | Last Name | string |  |
| 12 | Date | Date |  |
| 13 | Store Name | String |  |
| 14 | Store Number | String |  |
| 15 | Location | String | This is a string that uniquely identifies where the items are |
| 16 | Item Number | String |  |
| 17 | Description | String |  |
| 18 | Unit of Measure | String |  |
| 19 | Quantity on Hand | String |  |
| 20 | Quantity Counted | string |  |
| 21 | ABCD | string | classification |

### setErrorMessage

The purpose of this call resulted when a customer would get failures on performing shipping operations using the tablet mobile application and could not continue operations because an exception was thrown on the server. The mobile application would then retry and the server would continue to throw exceptions. In this particular case the exception was caused by having duplicated nodes for a part. When the server returns a 500 error to the mobile application the app will make this call to the server with the csv filename.

The web services will send an email to [roynabel@gmail.com](mailto:roynabel@gmail.com) with the filename and update a file \\servername\cdrive\UploadErrors\ErrorLog.txt.

For the setErrorMessage web service just update a file [\\servername\cdrive\UploadErrors\setErrorMessageLog.txt](file:///\\servername\cdrive\UploadErrors\setErrorMessageLog.txt) using the following line format.

Date, Time, Filename1

Date, Time, Filename2

Date, Time, Filename3

Date, Time, Filename4

Rand will update a file [\\servername\cdrive\UploadErrors\ErrorDetailsLog.txt](file:///\\servername\cdrive\UploadErrors\ErrorDetailsLog.txt) using the following line format.

Date, Time, Filename1, operation, itemNumber, serialnumber

Date, Time, Filename1, operation, itemNumber, serialnumber

Date, Time, Filename1, operation, itemNumber, serialnumber

Date, Time, Filename1, operation, itemNumber, serialnumber

When there are problems in processing a csv file the server will write the filename to the following folder and each line will correspond to the item that could not be processed.

[\\servername\cdrive\UploadErrors](file:///\\servername\cdrive\UploadErrors)

1611\_setShipParts\_shippingheader-1403222164762989\_2625.txt

**GET:**

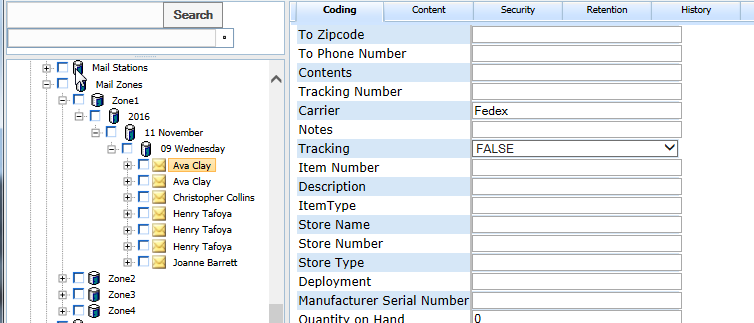
{webserver}/partservice/setErrorMessage/{login}/{password}/{filename}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| Filename | This is the csv file the mobile just uploaded that caused the error |

### setMailPackages

This call will create/update mail packages in a mail zone. If the mobilerecordid does not exist at the locationrecordid then you create the mail package in the mail zone. The directory structure looks like the following. You have a list of mail zones located under a storage node called Mail Zones. Within the Mail Zone first you break out the year, month and day and then you have a list of one or more mail packages. The mail package is a tracked record type and the quantity on hand is always 1.



**GET:**

{webserver}/partservice/setMailPackages/{login}/{password}/

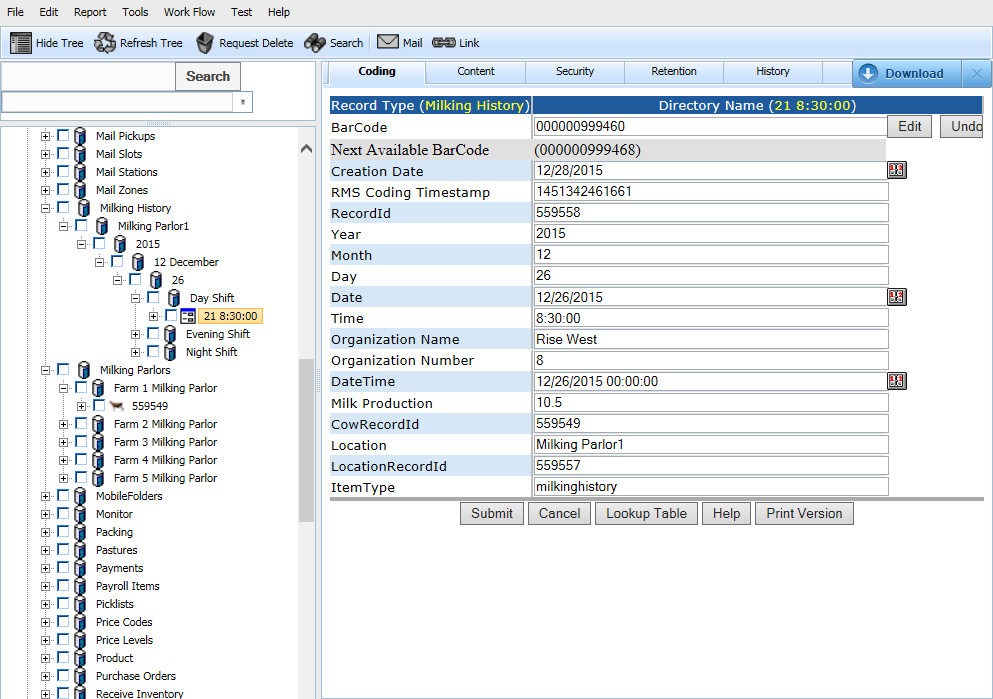
**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Datetime | string |  |
| 10 | Length | string |  |
| 11 | Width | string |  |
| 12 | Height | string |  |
| 13 | Weight | string |  |
| 14 | Location | string |  |
| 15 | Status | string |  |
| 16 | From Company | Date |  |
| 17 | From Name | Date |  |
| 18 | From Address | Integer |  |
| 19 | From City | Integer |  |
| 20 | From State | Integer |  |
| 21 | From Zipcode | String |  |
| 22 | From Phone Number | String |  |
| 23 | To Company | String |  |
| 24 | To Name | String |  |
| 25 | To Address | String |  |
| 26 | To City | String |  |
| 27 | To State | String |  |
| 28 | To Zipcode | String |  |
| 29 | To Phone Number | String |  |
| 30 | Contents | String |  |
| 31 | Tracking Number | String |  |
| 32 | Carrier | String | Logistics carrier = fedex, ups, usps, dhl |
| 33 | Notes | String |  |
| 34 | Tracking | String |  |
| 35 | Item Number | String |  |
| 36 | Description | String |  |
| 37 | ItemType | String |  |
| 38 | Store Name | String |  |
| 39 | Store Number | String |  |
| 40 | Store Type | String |  |
| 41 | Deployment | String |  |
| 42 | Manufacturer Serial Number | String |  |
| 43 | Quantity on Hand | Float |  |
| 44 | LocationRecordId | String |  |
| 45 | ContainerRecordId | String |  |
| 46 | Name | String |  |
| 47 | Direction | String | Mail flow direction = in or out |
| 48 | Mailcode | String |  |
| 49 | Container Type Name | String |  |
| 50 | Purchase Order | String |  |
| 51 | Vendor Name | String |  |
| 52 | Vendor Number | String |  |
| 53 | Sendor Name | String |  |
| 54 | Sendor Number | String |  |
| 55 | Cost Center | String |  |
| 56 | Resolution Notes | String |  |
| 57 | Destination Location | String |  |
| 58 | Special Handling | String |  |
| 59 | MailLabelRecordId | String |  |
| 60 | ContainerRecordId | string |  |
| 61 | Mail Zone | string |  |

### setMilkingHistories

This call will create/update milking histories. The directory structure looks like the following. The system will break out the miking history record first by location and then datetime. Note that in the database setting each organization can define the shift start and end times (24 hour format (zulu)).



**GET:**

{webserver}/partservice/setMilkingHistories/{login}/{password}/

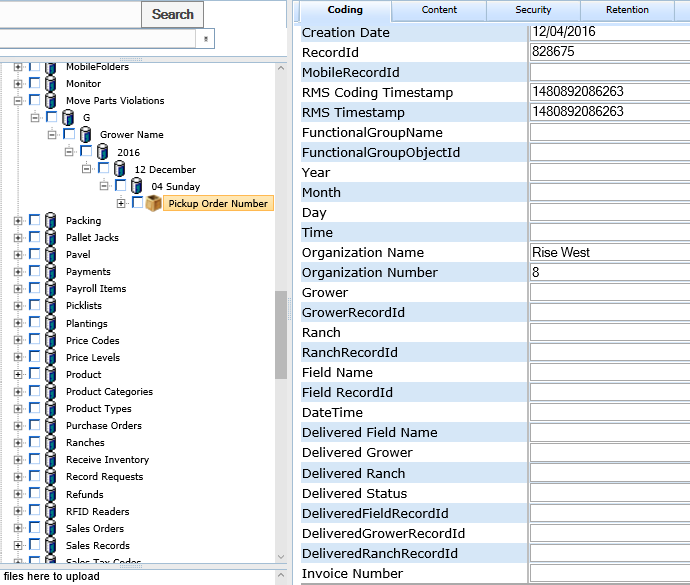
**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Datetime | string |  |
| 10 | Milk Production | string |  |
| 11 | CowRecordId | String |  |
| 12 | Location | String |  |
| 13 | LocationRecordId | String |  |
| 14 | ItemType | string |  |
| 15 | Farm Number | string |  |
| 16 | Farm Name | string |  |
| 17 | Shift Number | string |  |
| 18 | Shift Name | string |  |
| 19 | Device Id | String |  |
| 20 | Device Name | string |  |

### setMovePartsViolations

This keeps tracks of parts that were supposed to be in another location. The directory is broken down by the grower then the date and the violation is the pickup order number which is the invoice number.



**POST:**

{webserver}/partservice/movePartsViolations/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Grower | string |  |
| 10 | GrowerRecordId | string |  |
| 11 | Ranch | string |  |
| 12 | RanchRecordId | string |  |
| 13 | Field Name | string |  |
| 14 | FieldRecordId | string |  |
| 15 | DateTime | Date |  |
| 16 | Delivered Field Name | String |  |
| 17 | Delivered Grower | String |  |
| 18 | Delivered Ranch | String |  |
| 19 | Delivered Status | String |  |
| 20 | DeliveredGrowerRecordId | String |  |
| 21 | DeliveredRanchRecordId | String |  |
| 22 | Invoice Number | String |  |

### setPacking

This call is like the setShipParts except we add a layer for each truck since we are trying to have the shipping manifest for each truck and is a combination of createShipHeader and createShipDetail. After the header and detail are created the system will move the items to the destinationLocationRecordId.

Directory Structure

Packing

Year

Month

Day

Truck

Invoice Header

Invoice Detail

Invoice Detail

**GET:**

{webserver}/partservice/setPacking/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | OriginType | string | This can be INVOICE, CUSTOMERRMA, VENDORRMA |
| 10 | OriginNumber | string | This is the invoice#, CRMA#, VRMA# |
| 11 | MobileOriginNumber | string | If you generate the shipping offline (not recommended) |
| 12 | OriginName | string | This is the store name where the items are going |
| 13 | Carrier | string | Name of the shipping company |
| 14 | trackNumber | string | This is the shipping companies tracking id |
| 15 | destinationRecordId | string | this is the customer store receiving record id |
| 16 | Ship Date | Date | YYYY-MM-DD this the date items were sent to destination |
| 17 | Ship Time | Date | HH:MM:DD this is the time items were sent to destination |
| 18 | Number of Packages | Integer | Number of packages in shipment |
| 19 | SalesOrder | Integer | Sales order from accounting system |
| 20 | InvoiceNumber | Integer | Invoice Number from the accounting system |
| 21 | VehicleLicenseNumber | String | Truck vehicle license number |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a detail item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | ShippedItemRecordId | string | Part number can be non-tracked or tracked |
| 10 | Quantity | string | Number of items needed by receiver |
| 11 | Manufacturer Serial Number | string | This is the serial number that was scanned if you are not using recordId’s This was called ScanCode |
| 12 | FromLocationRecordId | string | This is where the part is coming from |
| 13 | Item Number Shipped | string | This is the part item number used to create a part with scancode when the part does not exist. |
| 14 | Item Number Received | string | This is the part item number that was received originally and may be changed in the item number shipped (ex a newer part) |
| 15 | Carrier | string | Shipping Carrier (fedex,dhl, ups, usps) |
| 16 | Tracking Number | string | This is the carrier’s shipping tracking number |
| 17 | Tracking Status | string | {delivered, transit, unknown} |
| 18 | Notes | string | Comments on a ship detail |
| 19 | Freight Amount | Fractional # | Freight amount for this shipment |
| 20 | Ship Zone | string | Where the shipment is going |
| 21 | Shipping Weight | Fractional # | Total weight of shipment |
| 22 | Number of Packages | Integer | Number of packages in shipment |
| 23 | Description | String | This is the item description |
| 24 | Invoice Number | String | This is from the accounting |

### setParts

This creates parts separated as follows in the directory. The record type is part.

**POST:**

{webserver}/partservice/setParts/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or Part Id will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | ObjectTypePartOverride | String |  |
| 10 | RecordPartType | String |  |
| 11 | LocationRecordId | string | This is the record id of the parent object where you want to store the part |
| 12 | LocationObjectId | string | This is the record id of the parent object where you want to store the part |
| 13 | LocationObjectType | string | This is the record id of the parent object where you want to store the part |
| 14 | Item Number | string | This is normally from the accounting system chart of accounts |
| 15 | Manufacturers Part Number | string | Note this only works with one vendor (lame) |
| 16 | Preferred Vendor | string | Note this only works with one vendor |
| 17 | Subitem of | string | This is when you group accounting items |
| 18 | Description on Purchase Transaction | string |  |
| 19 | Description on Sales Transaction | string |  |
| 20 | Cost | string | The amount you buy the part for from your vendor or distributor |
| 21 | Sales Price | string | The prices you sell the item to a customer |
| 22 | COGS Account | string | Cost of good sold account |
| 23 | Tax Code | string | This is for sales tax |
| 24 | Income Account | string | Which income account from the chart of accounts |
| 25 | Asset Account | string | If the item is purchased for internal use which chart of account |
| 26 | Reorder Point | string | This is the alert level when to reorder item |
| 27 | Quantity Picked | string |  |
| 28 | Quantity on Hand | string | This is how much of the item you really have if you did a hand count |
| 29 | Quantity on Hold | string | This is if somebody put a hold for a sales order |
| 30 | Quantity Counted | string | This is an inventory hand count |
| 31 | Total value | string | Not sure what people had in mind with this? |
| 32 | as of Date | string | This is a date to indicate when the coding values were applied |
| 33 | Deployment | string | Is the item in inventory, in transit, in the field |
| 34 | Location | String | A string to identify where the item is (record id should be used) Please don’t use a sequence! |
| 35 | Center Name | string |  |
| 36 | Center Number | string |  |
| 37 | Store Name | string |  |
| 38 | Store Number | Date |  |
| 39 | Aisle | String |  |
| 40 | Bay | String |  |
| 41 | Shelf | String |  |
| 42 | Customer RMA | string |  |
| 43 | Vendor RMA | string |  |
| 44 | Parent Id | string |  |
| 45 | Part Id | String |  |
| 46 | Tracking | String |  |
| 47 | Kit | String |  |
| 48 | Client Name | String |  |
| 49 | Client Number | String |  |
| 50 | U/M | String |  |
| 51 | Base U/M | String |  |
| 52 | Purchase U/M | String |  |
| 53 | Sales U/M | String |  |
| 54 | Ship Weight | String |  |
| 55 | Ship Via | String |  |
| 56 | Ship Date | String |  |
| 57 | Ship Time | String |  |
| 58 | Receive Date | String |  |
| 59 | Receive Time | string |  |
| 60 | Received By | String |  |
| 61 | Tracking Number | String |  |
| 62 | Master Barcode | String |  |
| 63 | Quantity Required | String |  |
| 64 | Required | String |  |
| 65 | Signature Required | String |  |
| 66 | ItemLabel | String |  |
| 67 | UserRecordId | String |  |
| 68 | Due Date | Date |  |
| 69 | Quantity Not Picked | string |  |
| 70 | Description | String |  |
| 71 | Alert Count | string |  |
| 72 | Alert Functional Group | string |  |
| 73 | Max Qty on Hand | string |  |
| 74 | Min Qty on Hand | string |  |
| 75 | Store Factor | string |  |
| 76 | Last Name | string |  |
| 77 | First Name | string |  |
| 78 | Record State | string |  |
| 79 | Record State Date | string |  |
| 80 | PTN | string |  |
| 81 | Model Number | string |  |
| 82 | ESN\_DEC | string |  |
| 83 | ESN\_HEX | string |  |
| 84 | MSL | string |  |
| 85 | MSID | string |  |
| 86 | Subscriber | String |  |
| 87 | Status | String |  |
| 88 | Contract Start | String |  |
| 89 | Contract End | String |  |
| 90 | Sims | String |  |
| 91 | IMEIs | String |  |
| 92 | PUK1 | String |  |
| 93 | PUK2 | string |  |
| 94 | VIN | string |  |
| 95 | Medallion Id | string |  |
| 96 | PIM Id | string |  |
| 97 | Vendor Name | string |  |
| 98 | Manufacturer Serial Number | string |  |

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setQuickBooksOnlineLog

This call is used to log the activity of the quickbooks online process. The system will build a folder for each date

QuickbooksOnlineLogs (folder name)

The following lines are in a file called qbolog20141217.txt

Date, Time, Message

Date, Time, Message

The following lines are in a file called qbolog20141218.txt

Date, Time, Message

Date, Time, Message

**GET:**

{webserver}/partservice/setQuickbooksOnlineLog/{login}/{password}/{message}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| Message | This is the csv string from quickbook online sync process |

### setPartsImport

This creates parts using the record type PartsImport and stored under a node in the Stores container called PartsImport with a store number 999999999999 (12 nines). Note that this creation uses all the fields. There is another part that is created in stored number and only two fields are used from the following csv (item number and description). This is done because when you attach a label to a part you can then scan the part record id and the manufacturer serial number you assign all the coding field you find by looking for the serial number in the PartsImport store. Once you assign the information you remove the PartImport record.

**POST:**

{webserver}/partservice/setPartsImport/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | LocationRecordId | string | This is the record id of the parent object where you want to store the part |
| 10 | LocationObjectId | string | This is the record id of the parent object where you want to store the part |
| 11 | LocationObjectType | string | This is the record id of the parent object where you want to store the part |
| 12 | Item Number | string | This is normally from the accounting system chart of accounts |
| 13 | Manufacturers Part Number | string | Note this only works with one vendor (lame) |
| 14 | Preferred Vendor | string | Note this only works with one vendor |
| 15 | Subitem of | string | This is when you group accounting items |
| 16 | Description on Purchase Transaction | string |  |
| 17 | Description on Sales Transaction | string |  |
| 18 | Cost | string | The amount you buy the part for from your vendor or distributor |
| 19 | Sales Price | string | The prices you sell the item to a customer |
| 20 | COGS Account | string | Cost of good sold account |
| 21 | Tax Code | string | This is for sales tax |
| 22 | Income Account | string | Which income account from the chart of accounts |
| 23 | Asset Account | string | If the item is purchased for internal use which chart of account |
| 24 | Reorder Point | string | This is the alert level when to reorder item |
| 25 | Quantity Picked | string |  |
| 26 | Quantity on Hand | string | This is how much of the item you really have if you did a hand count |
| 27 | Quantity on Hold | string | This is if somebody put a hold for a sales order |
| 28 | Quantity Counted | string | This is an inventory hand count |
| 29 | Total value | string | Not sure what people had in mind with this? |
| 30 | as of Date | string | This is a date to indicate when the coding values were applied |
| 31 | Deployment | string | Is the item in inventory, in transit, in the field |
| 32 | Location | String | A string to identify where the item is (record id should be used) Please don’t use a sequence! |
| 33 | Center Name | string |  |
| 34 | Center Number | string |  |
| 35 | Store Name | string |  |
| 36 | Store Number | Date |  |
| 37 | Aisle | String |  |
| 38 | Bay | String |  |
| 39 | Shelf | String |  |
| 40 | Customer RMA | string |  |
| 41 | Vendor RMA | string |  |
| 42 | Parent Id | string |  |
| 43 | Part Id | String |  |
| 44 | Tracking | String |  |
| 45 | Kit | String |  |
| 46 | Client Name | String |  |
| 47 | Client Number | String |  |
| 48 | U/M | String |  |
| 49 | Base U/M | String |  |
| 50 | Purchase U/M | String |  |
| 51 | Sales U/M | String |  |
| 52 | Ship Weight | String |  |
| 53 | Ship Via | String |  |
| 54 | Ship Date | String |  |
| 55 | Ship Time | String |  |
| 56 | Receive Date | String |  |
| 57 | Receive Time | string |  |
| 58 | Received By | String |  |
| 59 | Tracking Number | String |  |
| 60 | Master Barcode | String |  |
| 61 | Quantity Required | String |  |
| 62 | Required | String |  |
| 63 | Signature Required | String |  |
| 64 | ItemLabel | String |  |
| 65 | UserRecordId | String |  |
| 66 | Due Date | Date |  |
| 67 | Quantity Not Picked | string |  |
| 68 | Description | String |  |
| 69 | Alert Count | string |  |
| 70 | Alert Functional Group | string |  |
| 71 | Max Qty on Hand | string |  |
| 72 | Min Qty on Hand | string |  |
| 73 | Store Factor | string |  |
| 74 | Last Name | string |  |
| 75 | First Name | string |  |
| 76 | Record State | string |  |
| 77 | Record State Date | string |  |
| 78 | PTN | string |  |
| 79 | Model Number | string |  |
| 80 | ESN\_DEC | string |  |
| 81 | ESN\_HEX | string |  |
| 82 | MSL | string |  |
| 83 | MSID | string |  |
| 84 | Subscriber | String |  |
| 85 | Status | String |  |
| 86 | Contract Start | String |  |
| 87 | Contract End | String |  |
| 88 | Sims | String |  |
| 89 | IMEIs | String |  |
| 90 | PUK1 | String |  |
| 91 | PUK2 | string |  |
| 92 | VIN | string |  |
| 93 | Medallion Id | string |  |
| 94 | PIM Id | string |  |
| 95 | Vendor Name | string |  |

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setReceiveParts

This call is a combination of receivePartHeader and receivePartDetail. The reason why we added this call was for speed.

**GET:**

{webserver}/partservice/setReceiveParts/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
|  | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | origin type | string | This is PURHASE, CUSTOMERRMA, VENDORRMA |
| 10 | origin number | string | This is generated by the mobile device or the accounting system if attached |
| 11 | Mobile Origin Number | string | The mobile device generates a purchase order number for new orders, or uses customer rma number for a customer return or vendor rma number for a vendor return |
| 12 | origin name | string | This is the store name from where you are receiving items: for a purchase then you have the vendor store name and also for a vendor rma. For a customer return you will a customer store name |
| 13 | receiptDate | string | This is when you received the items |
| 14 | referenceNumber | string | This is from the vendor identifying the po |
| 15 | Processed | string | Yes or no |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a detail item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | OriginType | string | This is PURHASE, CUSTOMERRMA, VENDORRMA |
| 10 | OriginNumber | string |  |
| 11 | MobileOriginNumber | string | The mobile device generates a purchase order number for new orders, or uses customer rma number for a customer return or vendor rma number for a vendor return |
| 12 | OriginName | string | This is the store name from where you are receiving items: for a purchase then you have the vendor store name and also for a vendor rma. For a customer return you will a customer store name |
| 13 | receiptDate | string | This is when you received the items |
| 14 | Scan Code | string | This is the manufacturer serial number |
| 15 | \*\*ItemNumber or itemLabel | string | This is the item number from the chart of accounts |
| 16 | itemDescription | string | Description of item received |
| 17 | ManufacturerSerialNumber | string | This is when you received the items |
| 18 | Quantity | string | This is when you received the items |
| 19 | OldLocationRecordId | string | This is where the received items will come from. This is mainly used for non-tracked items so you can set quantities |
| 20 | DestinationLocationRecordId | string | This is where the received items will get created |
| 21 | Processed | string | Yes or no |
| 22 | UnitOfMeasure | string |  |
| 23 | Cost | string |  |
| 24 | Amount | string |  |
| 25 | Billable | Boolean |  |
| 26 | Carrier | string | Shipping Carrier (fedex,dhl, ups, usps) |
| 27 | Tracking Number | string | This is the carrier’s shipping tracking number |
| 28 | Tracking Status | string | {delivered, transit, unknown} |
| 29 | Notes | string | Comments on line detail |
| 30 | ItemMobileRecordId | String | When you receive an item into a store then that item get created with this mobile recordid |

### setReceivePartsSingle

This call will create a single receiver part (header and detail). Please see the call setReceiveParts for a description of the header and detail. The reason why we added this call was because the ios and android communication libraries don’t do good handshaking and the csv file does not get attached to the web service call when received by tomcat. We add a second retry but sometimes this is not enough so we increased the retries to 3 times and then the single record is called until all records get processed. Note that there are 3 retries on this call and if the call fails 3 times the next record is processed. At the end if there are more than 0 errors a message is displayed to the operator to contact support and the number of errors.

**GET:**

{webserver}/partservice/setReceivePartsSingle/{login}/{password}/{header}/{detail}/

### setFieldBoxes

This call creates and/or updates a field box which is its own record type with coding fields specific to shipping since we do not want to overload a part’s coding fields. Note that you receive shipping boxes as non-tracked items and then you create a product box and decrement the non-track part item quantity on hand in the fromLocationRecordId. The product box will be created under the node called Product and will have a node name specified by Field Box Name (coding field see below).

**GET:**

{webserver}/partservice/setFieldBoxes/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Box Number | string | This is the non-track part of a product box |
| 10 | Item Number | string | This is the non-track part of a product box |
| 11 | Lot | string | This is the lot where the raw product came from |
| 12 | Batch | string | This is the batch where the product came from |
| 13 | FieldBoxRecordId | string | This is the store where the product came from. Normally this is where you receive inventory |
| 14 | FieldWorkers | string | Name of worker(s) |
| 15 | FieldSection | string | Section where field was workd |
| 16 | LoadDateTime | datetime | Date/time field box was loaded |
| 17 | Product Type | string | Short description of finished product rasberry |
| 16 | FieldBoxName | string | Directory Box Number (assigned by rms) Product Type |

### fillProducePacks

This call updates the coding fields for multiple produce packs. Note that coding ids have to be unique within the entire organization and all children sub-organizations.

**GET:**

{webserver}/partservice/fillProducePacks/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | NewItemNumber | String | Based on the accounting item number so if you have an empty product case and you scan strawberries into the product pack now the item is a strawberry pack but the record type is still a product pack |
| 10 | NewItemDesciption | String | This is the retail package description |
| 11 | Datetime | String | Ansi format YYYY-MM-DD HH:MM:SS.SSS |
| 12 | Produce Pack Manufacturer Serial Number | String | This is the plastic clamshell |
| 13 | Container Manufacturer Serial Number | String | This is the tray |
| 14 | Grower | string | This is the id of the grower |
| 15 | Ranch | string | This is the ranch id since grower can have multiple locations |
| 16 | Block | String | This is the block id within a given ranch |
| 17 | Row | string | This is the field box where the product came from |
| 18 | UserRecordId | string | This is the recordid of the person that packed the produce pack |
| 19 | VehicleLicenseNumber | string | this is the truck vehicle license number must be unique world wide |
| 20 | Vehicle Manufacturer Serial Number | string | This is the truck manufacturer serial number |
| 21 | Quantity on Hand | string | Always 1 |
| 22 | Deployment | string | Always inventory |

### setForklifts

This creates edits the forklift record type.

**GET:**

{webserver}/partservice/setForklifts/{login}/{password}/

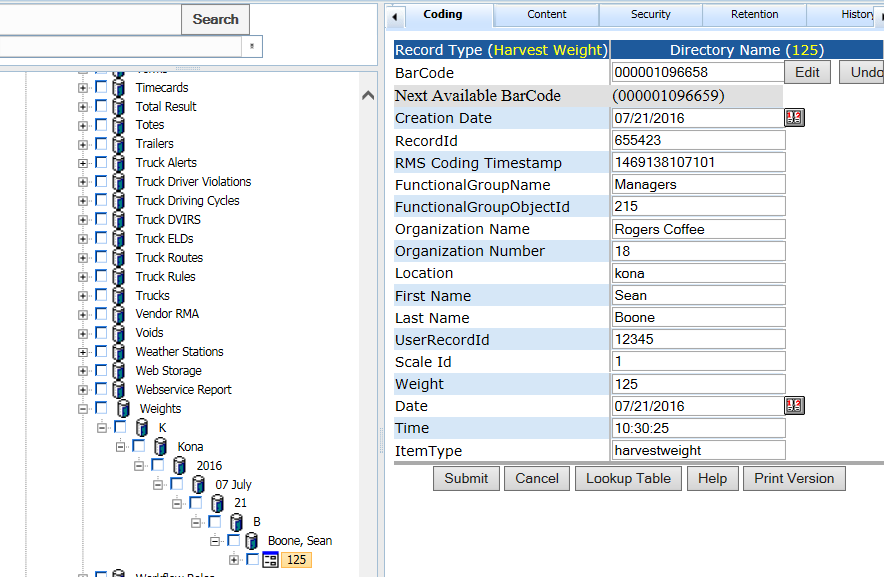
**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Number | string |  |
| 10 | Purchase Date | string |  |
| 11 | Last Worked | string |  |
| 12 | ItemType | string |  |

### setHarvestWeight

This creates weight measure as shown in the directory figure below. Use the first letter of the location then the location name then year, month day and then first letter of last name. The record name is just the weight. Note the system will look up the userrecordid and get the first name, last name and location.



**GET:**

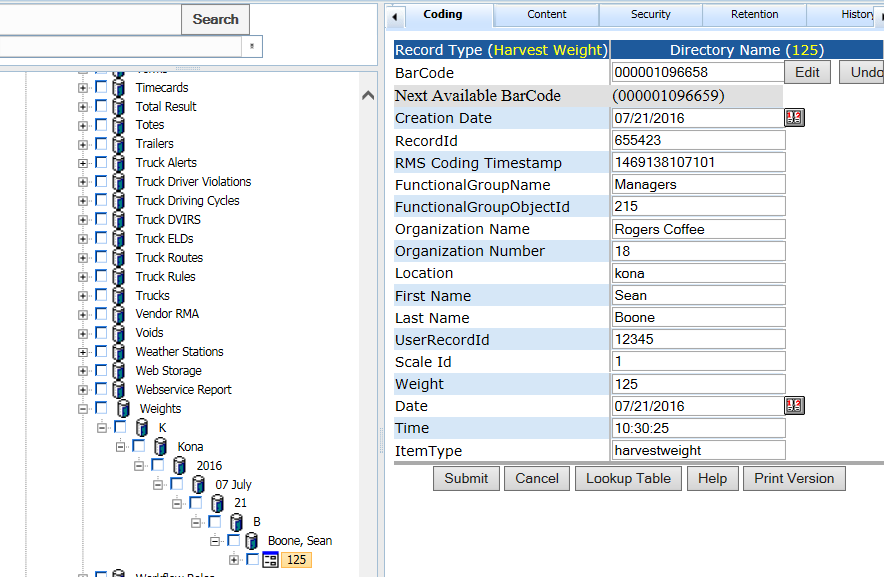
{webserver}/partservice/setHarvestWeight/{login}/{password}/{DateTime}/{UserRecordId}/{Scale Id}/{Weight}/{ItemType}/{Location}/{FieldName}/{Product Category}/{Product Type}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| DateTime | This is the date weight was done YYYYMMDD HHMMSS |
| UserRecordId | This is user who produced product getting weighed |
| Scale Id | This is a unique scale id for entire organization |
| Weight | This is the weight in pounds |
| ItemType | “harvestweight” used for filtering operations |
| Location | This is the geographic area (ranch) where weight recorded |
| FieldName | This is the unique field where the weight recorded |
| Product Category | This a general class describing the product |
| Product Type | This is a detailed class of product type |

### setHarvestWeights

This creates weight measure as shown in the directory figure below. Use the first letter of the location then the location name then year, month day and then first letter of last name. The record name is just the weight. Note the system will look up the userrecordid and get the first name, last name and location.



**GET:**

{webserver}/partservice/setHarvestWeights/{login}/{password}/

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | DateTime | string | This is the date weight was done YYYYMMDD HHMMSS |
| 10 | UserRecordId | String | This is user who produced product getting weighed |
| 11 | Scale Id | String | This is a unique scale id for entire organization |
| 12 | Weight | String | This is the weight in pounds |
| 13 | ItemType | String | “harvestweight” used for filtering operations |
| 14 | Location | String | This is the geographic area (ranch) where weight recorded |
| 15 | Product Category | String | This a general class describing the product |
| 16 | Product Type | String | This is a detailed class of product type |
| 17 | Estimated Yield | String |  |
| 18 | Status | String |  |
| 19 | Field Name | String |  |
| 20 | Field Number | String |  |
| 21 | Container Type | String |  |
| 22 | Container Number | String |  |
| 23 | Harvest Area Name | String |  |
| 24 | Harvest Area Number | String |  |
| 25 | Acres | string |  |

### setPalletJacks

This creates edits the pallet jack record type

**GET:**

{webserver}/partservice/setPalletJacks/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Number | string |  |
| 10 | Purchase Date | string |  |
| 11 | Last Worked | string |  |
| 12 | ItemType | string |  |

### setServiceTickets

This function is used to create/modify/delete service tickets in the directory for a given organization and to set the coding fields using a CSV file.The data for the csv file will normally come from the mobile device used by the service manager. In the directory service tickets are stored under a node called “Service Tickets” using the YYYY/MM/DD format. The directory node name is formed by concatenating the recordId of the node – Medallion Id. The directory node is of “Service Ticket” record type.

**GET:**

{webserver}/partservice/setServiceTickets/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Organization Name | String |  |
| 8 | Organization Number | string |  |
| 9 | Arrival Date | Date | mm/dd/yyyy |
| 10 | Arrival Time | string | hh:mm:ss (24 hour mode) - when the vehicle arrived |
| 11 | Medallion Id | string | This is the vehicle medallion number |
| 12 | Technician1 | String | Primary technician |
| 13 | Technician2 | String | Secondary technician |
| 14 | Service Type | string | This is the service performed on the vehicle |
| 15 | Bad Parts | string | This is a string of part record id’s separated by a space |
| 16 | Description | string | This is a general short version of the service done on vehicle |
| 17 | Replacement Parts | string | This is a string of part record id’s separated by a space |
| 18 | Departure Time | String | hh:mm:ss (24 hour mode) - when the vehicle left |
| 19 | Bad Parts Serial Numbers | String | These are the serial numbers associated with the record id’s in the Bad Parts coding fields |
| 20 | Replacement Parts Serial Numbers | String | These are the serial numbers associated with the record id’s in the Replacement Parts coding fields |
| 21 | Bad Parts RecordIds | String | Look at Item Primary Key Coding Field on the Database record type to determine whether you are scanning recordIds or manufacturer serial numbers into this field |
| 22 | Replacement Parts RecordIds | String | Look at Item Primary Key Coding Field on the Database record type to determine whether you are scanning recordIds or manufacturer serial numbers into this field |

### setServiceTicketsSingle

This call process a single ServiceTicket header and detail. Please see setServiceTickets for a description of the header and detail. The reason why we added this call was because the ios and android communication libraries don’t do good handshaking and the csv file does not get attached to the web service call when received by tomcat. We add a second retry but sometimes this is not enough so we increased the retries to 3 times and then the single record is called until all records get processed. Note that there are 3 retries on this call and if the call fails 3 times the next record is processed. At the end if there are more than 0 errors a message is displayed to the operator to contact support and the number of errors.

**GET:**

{webserver}/partservice/setServiceTicketsSingle/{login}/{password}/{header}/{detail}

### setShippingBoxes

This call creates and/or updates a shipping box which is its own record type with coding fields specific to shipping since we do not want to overload a part’s coding fields. Note that you receive shipping boxes as non-tracked items and then you create a shipping box and decrement the non-track part item quantity on hand in the fromLocationRecordId. The shipping box will be created under the node called Packing and will have a node name specified by Shipping Box Name (coding field see below).

**GET:**

{webserver}/partservice/setShippingBoxes/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Item Number | string | This is the non-track part of a shipping box |
| 10 | FromLocationRecordId | string | This is the store record id where we take the non-track box |
| 11 | Carrier | string | This is the shipping carrier |
| 12 | Shpping Weight | string | This is the weight of the give box |
| 13 | Tracking Number | string | This is the label supplied by the shipping carrier |
| 14 | Number of Boxes | string | this is the total number of boxes to be shipped |
| 15 | Series Number | string | This is box X of total number of boxes |
| 16 | Serial Numbers | string | This is a comma separated list of all the boxes |
| 17 | Shipping Box Name | string | This is the directory node name invoice number, customer name, series number |
| 18 | Status | string | Not shipped, In Transit, Lost, Delivered |
| 19 | Delivery Date | String | Date when the box was delivered |
| 20 | Delivery Time | String | Time when the box was delivered |
| 21 | Received By | String | Person who received the box |

### setShipParts

This call is a combination of createShipHeader and createShipDetail. After the header and detail are created the system will move the items to the destinationLocationRecordId.

**GET:**

{webserver}/partservice/setShipParts/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | OriginType | string | This can be INVOICE, CUSTOMERRMA, VENDORRMA |
| 10 | OriginNumber | string | This is the invoice#, CRMA#, VRMA# |
| 11 | MobileOriginNumber | string | If you generate the shipping offline (not recommended) |
| 12 | OriginName | string | This is the store name where the items are going |
| 13 | Carrier | string | Name of the shipping company |
| 14 | trackNumber | string | This is the shipping companies tracking id |
| 15 | destinationRecordId | string | this is the customer store receiving record id |
| 16 | mobileRecordId | string | This is unique identifier created by mobile device |
| 17 | Ship Date | Date | YYYY-MM-DD this the date items were sent to destination |
| 18 | Ship Time | Date | HH:MM:DD this is the time items were sent to destination |
| 19 | Number of Packages | Integer | Number of packages in shipment |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a detail item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | shippedItemRecordId | string | Part number can be non-tracked or tracked |
| 10 | quantity | string | Number of items needed by receiver |
| 11 | ScanCode | string | This is the serial number that was scanned if you are not using recordId’s |
| 12 | FromLocationRecordId | string | This is where the part is coming from |
| 13 | Item Number Shipped | string | This is the part item number used to create a part with scancode when the part does not exist. |
| 14 | Item Number Received | string | This is the part item number that was received originally and may be changed in the item number shipped (ex a newer part) |
| 15 | Carrier | string | Shipping Carrier (fedex,dhl, ups, usps) |
| 16 | Tracking Number | string | This is the carrier’s shipping tracking number |
| 17 | Tracking Status | string | {delivered, transit, unknown} |
| 18 | Notes | string | Comments on a ship detail |
| 19 | Freight Amount | Fractional # | Freight amount for this shipment |
| 20 | Ship Zone | string | Where the shipment is going |
| 21 | Ship Weight | Fractional # | Total weight of shipment |
| 22 | Number of Packages | Integer | Number of packages in shipment |

### setShipPartsNoReturnData

This call is the same as setShipParts except that it only returns a message. The purpose of this call is used when importing old data and preserving the shipping dates of items.

**GET:**

{webserver}/partservice/setShipPartsNoReturnData/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | OriginType | string | This can be INVOICE, CUSTOMERRMA, VENDORRMA |
| 10 | OriginNumber | string | This is the invoice#, CRMA#, VRMA# |
| 11 | MobileOriginNumber | string |  |
| 12 | OriginName | string |  |
| 13 | Carrier | string | Name of the shipping company |
| 14 | trackNumber | string | This is the shipping companies tracking id |
| 15 | destinationRecordId | string | this is the customer store receiving record id |
| 16 | mobileRecordId | string | This is unique identifier created by mobile device |
| 17 | Ship Date | Date | YYYY-MM-DD this the date items were sent to destination |
| 18 | Ship Time | Date | HH:MM:DD this is the time items were sent to destination |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a detail item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | RecordId | string | Part number can be non-tracked or tracked |
| 10 | quantity | string | Number of items needed by receiver |
| 11 | ScanCode | string | This is the serial number that was scanned if you are not using recordId’s |
| 12 | FromLocationRecordId | string | This is where the part is coming from |
| 13 | Item Number | string | This is the part item number used to create a part with scancode when the part does not exist. |
| 14 | Carrier | string | Shipping Carrier (fedex,dhl, ups, usps) |
| 15 | Tracking Number | string | This is the carrier’s shipping tracking number |
| 16 | Tracking Status | string | {delivered, transit, unknown} |
| 17 | Notes | string | Comments on a ship detail |

### setShipPartsSingle

This call does a single header and detail. Please see setShipParts for a description of the header and detail records. The reason why we added this call was because the ios and android communication libraries don’t do good handshaking and the csv file does not get attached to the web service call when received by tomcat. We add a second retry but sometimes this is not enough so we increased the retries to 3 times and then the single record is called until all records get processed. Note that there are 3 retries on this call and if the call fails 3 times the next record is processed. At the end if there are more than 0 errors a message is displayed to the operator to contact support and the number of errors.

**GET:**

{webserver}/partservice/setShipPartsSingle/{login}/{password}/{header}/{detail}

### setShips

This creates ships separated as follows in the directory. You take the first letter of the name of the ship. The record type is Ship.

Ships

K

Kodiak Enterprise

**POST:**

{webserver}/partservice/setShips/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Type | string | Type of vessel |
| 10 | Name | Int | Name of vessel |
| 11 | Number of Crew | Int |  |
| 12 | Species | String | Type of fish caught |
| 13 | Length | Int | Overall length of vessel |
| 14 | Breadth | Int |  |
| 15 | Draught | Int |  |
| 16 | Gross Tons | Int |  |
| 17 | Horse Power | Int |  |
| 18 | Mains | String |  |
| 19 | Built | String | Year vessel was first built |
| 20 | Refrigeratuion | String |  |
| 21 | First Name | String |  |
| 22 | Last Name | String |  |
| 23 | MobilePhone | String |  |
| 24 | Overdue | String |  |
| 25 | Overdue Time Limit | String |  |
| 26 | Active | String |  |
| 27 | Flag | String |  |
| 28 | IMO | String |  |
| 29 | MMSI | String |  |
| 30 | Call Sign | String |  |
| 31 | Cargo | String |  |
| 32 | Latitude | String |  |
| 33 | Longitude | String |  |
| 34 | Heading | float |  |
| 35 | Course |  |  |
| 36 | Speed | Float |  |
| 37 | Source |  |  |
| 38 | Rate of turn |  |  |
| 39 | Destination |  |  |
| 40 | Estimated Time of Arrival |  |  |

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setStores

This function is used to create/modify/delete stores in the directory for a give organization and to set the coding fields using a CSV file.The data for the csv file will normally come from the accounting system. This function is useful to configure the rms and is used to move return parts back to a vendor. When creating a store the rms will look for the maximum store number string for a given login organization and assign the new store number = maximum store number + 1

Library = -1

**POST:**

{webserver}/partservice/setStores/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Organization Name | String |  |
| 8 | Organization Number | string |  |
| 9 | Store Name | string |  |
| 10 | Store Number | string |  |
| 11 | Store Id | String |  |
| 12 | Store Type | String |  |
| 13 | Center Name | string |  |
| 14 | Center Number | string |  |
| 15 | Company | string |  |
| 16 | First Name | string |  |
| 17 | Last Name | String |  |
| 18 | Address1 | String |  |
| 19 | City | String |  |
| 20 | State | String |  |
| 21 | ZipCode | String |  |
| 22 | Country | String |  |
| 23 | Phone | String |  |
| 24 | Deployment | String |  |
| 25 | Location | String |  |
| 26 | IsaLocation | Boolean |  |
| 27 | Store Factor | string |  |
| 28 | Alert Functional Group | string |  |
| 29 | Directory Style | string |  |
| 30 | Latitude | string |  |
| 31 | Longitude | string |  |
| 32 | OwnerRecordId | string | This is the record id of the store owner which is the record id of some user. |
| 33 | Temperature1 | string | Temperature in store |
| 34 | Store Class | String | This describes the general type of store warehouse, coldstorage, computers |

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setSyncMessage

This call logs in the sync status so we can determine which record types take too long. The web service will add a carriage return linefeed after each call.

The web services will update a file \\servername\cdrive\UploadErrors\SyncLog.txt.

Date, Time, SyncDescripiton

Date, Time, SyncDescripiton

Number of Records to Update, Duration (sec)

**GET:**

{webserver}/partservice/setSyncLog/{login}/{password}/{SyncDescription}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| SyncDescription | This is a string with sync status |

### setTaxis

This creates vehicles separated as follows in the directory. Note that you may not have a State in all cases. The record type is Taxi.

Taxis

Country

State or Region

City

Operator Name

Vehicle Identification Number

Part1

Part2

**POST:**

{webserver}/partservice/setTaxis/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

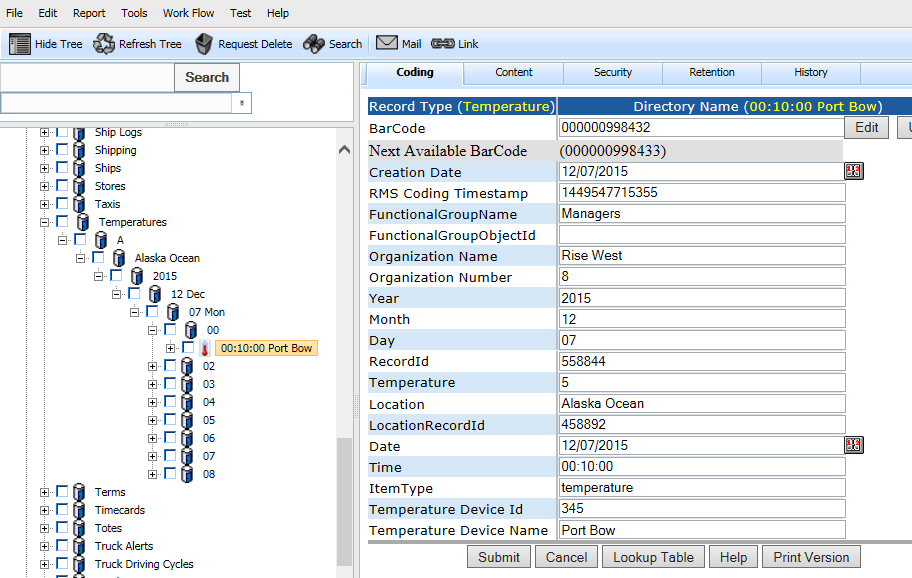
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Medallion Id | string | This is specific to the new York area |
| 8 | PIM Id | string |  |
| 9 | VIN | string |  |
| 10 | Vehicle License Number | string |  |
| 11 | Customer Number | string |  |
| 12 | Operator Name | string | Name of the customers |
| 13 | House Number | string |  |
| 14 | Make | string |  |
| 15 | Model | string |  |
| 16 | Year | string |  |
| 17 | Installed Status | string |  |
| 18 | Vehicle Status | string |  |
| 19 | Medallion Status | string |  |
| 20 | Install Date | string |  |
| 21 | Mac Address | string |  |
| 22 | SSID | string |  |
| 23 | ESN | string |  |
| 24 | Build Version | string |  |
| 25 | OS | string |  |
| 26 | EID | string |  |
| 27 | Modem Phone Number | string |  |
| 28 | MDT Version | string |  |
| 29 | Dispatch Status | string |  |
| 30 | Manufacturer | string |  |
| 31 | Last Worked | Date |  |
| 32 | Country | String |  |
| 33 | StateRegion | String |  |
| 34 | City | String |  |
| 35 | Organization Name | String |  |
| 36 | Organization Number | String |  |

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setTemperatures

This creates a temperature measure record as shown in the directory figure below. Use the first letter of the location then the location name then year, month, day and hour and then the temperature. Time is in 24 hour format.



**POST:**

{webserver}/partservice/setTemperatures/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Temperature | Float | temperature in celcius |
| 10 | Location | string | General location name of temperature device |
| 11 | LocationRecordId | string |  |
| 12 | Date | Date |  |
| 13 | Time | String |  |
| 14 | ItemType | string | Temperature |
| 15 | Temperature Device Id | string | Every weight scale in organization needs a unique scale id |
| 16 | Temperature Device Name | string | This is a relevant scale name |

### setTrailers

This creates trailers separated as follows in the directory. Note that you may not have a State in all cases. The record type is Trailer.

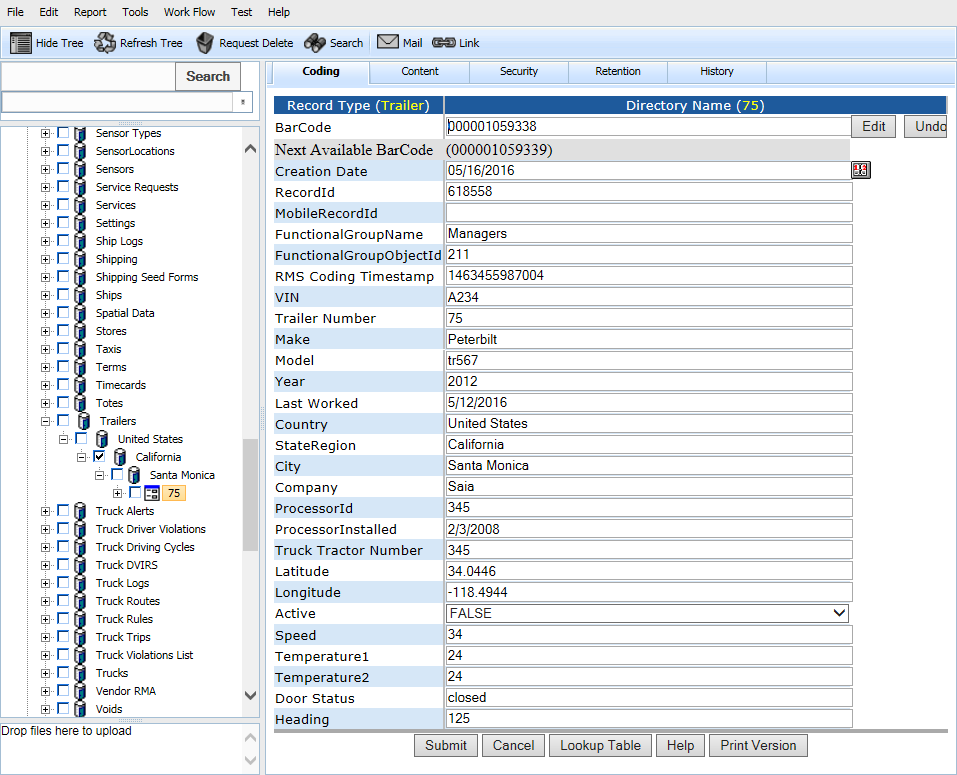
Trailers

Country

State or Region

City

TrailerNumber



**POST:**

{webserver}/partservice/setTrailers/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | VIN | string | This is the trailer vehicle identification number |
| 10 | Trailer Number | string |  |
| 11 | Make | string |  |
| 12 | Model | string |  |
| 13 | Year | string |  |
| 14 | Last Worked | Date |  |
| 15 | Country | String |  |
| 16 | StateRegion | String |  |
| 17 | City | String |  |
| 18 | Company | String |  |
| 19 | ProcessorId | String |  |
| 20 | ProcssorInstalled | String |  |
| 21 | Truck Tractor Number | String |  |
| 22 | Latitude | String |  |
| 23 | Longitude | String |  |
| 24 | Active | Boolean |  |
| 25 | Speed | String |  |
| 26 | Temperature1 | String |  |
| 27 | Temperature2 | String |  |
| 28 | Door Status | String |  |
| 29 | Heading | String |  |
| 30 | Generator Hours | String |  |
| 31 | Miles | String |  |
| 32 | Fuel Rate | String |  |
| 33 | Shock | String |  |
| 34 | Tire Pressures | String |  |
| 35 | ItemType | String |  |
| 36 | Treads | string |  |

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setTrucks

This creates vehicles separated as follows in the directory. Note that you may not have a State in all cases. The record type is Truck.

Trucks

Country

State or Region

City

LicenseNumber

Part1

Part2

**POST:**

{webserver}/partservice/setTrucks/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

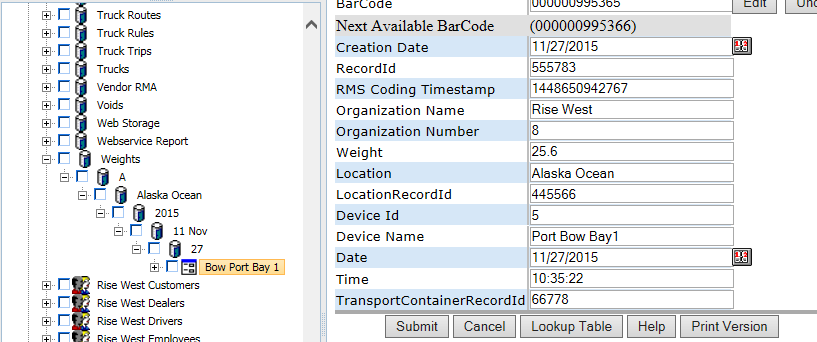
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | VIN | string | This is the vehicle license |
| 10 | Vehicle License Number | string |  |
| 11 | Make | string |  |
| 12 | Model | string |  |
| 13 | Year | string |  |
| 14 | Last Worked | Date |  |
| 15 | Country | String |  |
| 16 | StateRegion | String |  |
| 17 | City | String |  |
| 18 | Company | String |  |
| 19 | First Name | String |  |
| 20 | Last Name | String |  |
| 21 | MobilePhone | String |  |
| 22 | Latitude | String |  |
| 23 | Longitude | String |  |
| 24 | Active | Boolean |  |
| 25 | Speed | String |  |
| 26 | Temperature1 | String |  |
| 27 | Temperature2 | String |  |
| 28 | Door Status | String |  |
| 29 | Heading | String |  |
| 30 | Truck Number | String |  |
| 31 | Odometer | Int |  |
| 32 | EngineIdle | Bool |  |
| 33 | RPM | int |  |
| 34 | Engine Hours | int |  |
| 35 | Odometer Setup | string |  |
| 36 | Shock | string |  |

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setWeights

This creates weight measure as shown in the directory figure below. Use the first letter of the location then the location name then year, month day and then the weight.



**POST:**

{webserver}/partservice/setWeights/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Weight | Float | Weight in kilograms |
| 10 | Location | string | General location name of weight scales |
| 11 | LocationRecordId | string |  |
| 12 | Date | Date |  |
| 13 | Time | String |  |
| 14 | TransportContainerRecordId | String |  |
| 15 | ItemType | string | weight |
| 16 | Scale Id | string | Every weight scale in organization needs a unique scale id |
| 17 | Scale Name | string | This is a relevant scale name |
| 18 | Process Name | string | reject |
| 19 | Product | String | Lemon |

### setupPart

This creates a single part in the library store which is hard coded as a constant in the code and you use for right the setRecordCoding api to assign values.

**POST:**

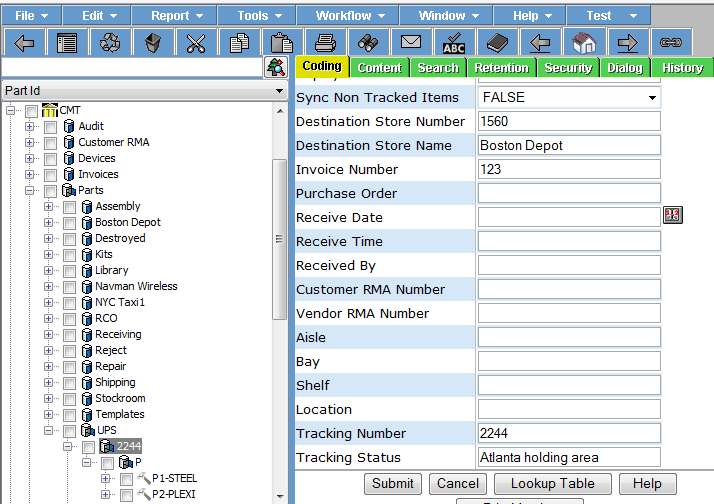
{webserver}/partservice/setupPart/{login}/{password}/{itemNumber}/{quantity}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| itemNumber | Is the part number from the accounting system |
| Quantity | Added this for the audit log. If parts exists then rms only updates the quantity. |

### shipPart

This restful interface is used to ship a part to a location. The ship part moves a part to a store where the store name and number are the tracking number. If the store number does not exist then the method creates the store and adds the child record and sets the coding fields.



The following code describes the shipPart function.

If InvoiceNumber is not blank then we have a ship to customer

If store.tracking number does not exist then create store.tracking number

createPart

lookup customerName from invoiceNumber

lookup destinationLocation from user.CustomerNumber

set Destination StoreNumber, Store Number and Location

If VendorRma is not blank then we have a ship to vendor

If store.tracking number does not exist then create store.tracking number

createPart

lookup location from vendorRmaNumber

set Destination StoreNumber, Store Number and Location

The following describes the tracking of part. There is a timer process that runs on the web server machine that triggers a method to check all the tracking numbers that have not been received. When a tracking number is delivered the children of the store are moved to the destination store number.

getActiveTrackingIdsAll – return id if deployment = transit and received date = null

timer with Cygwin (installed on rms server where apache web server is running)

timer interval = 1 hr

deployment = depot

loop through all ids

call ups server with tracking number and update the coding fields

received date

received time

received by

tracking status

set deployment to delivered

move all children of the id and update deployment per config file parameter

**POST:**

{webserver}/partservice/shipPart/{login}/{password}/{invoiceNumber}/{vendorRma}/{carrier}/{trackingId}/{itemRecordId}/{quantity}/{newLocationRecordId}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| InvoiceNumber | Invoice number from accounting system |
| vendorRma | This is the number supplied by the vendor |
| Tracking id | This is the shippers tracking id |
| itemRecordId | This is the part you are receiving |
| Quantity | Number of items to ship |
| newLocationRecordId | This is where part is new location of part |

### updateSerialNumbers

Update a given list of existing parts given a csv file (see next table)

**POST:**

{webserver}/partservice/updateSerialNumbers/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | partRecordId | string | This is the record id of the part you are moving |
| 2 | SerialNumber | string | Number of parts to move. A tracked part must have a quantity of 1 |

## Record Service

This section deals with all the operations that work on a database record set constrained by function arguments.

### appendRecordContent

This call appends the efile content for a given directory id. When we wrote this function this is a non-encrypted csv file that records truck position.

Date,time,vechicle license number,latitude,longitude,speed,heading

**POST:**

{webserver}/recordservice/appendRecordContent/{login}/{password}/{objectId}/{objectType}{key}{Mode}

**EXAMPLE:**

curl -k -X POST -F [media=@6441.txt](mailto:media=@6441.txt) {webserver}/recordservice/appendRecordContent/login/password/6440/User Note that 6441.txt is the file to be uploaded to the server.

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| objectId | Record id |
| objectType | Record type |
| Key | Encryption key for server. If blank then no encryption else aes256 |
| Mode | Encryption mode for server.  0=no encryption  1=use password for encryption  2=use supplied key string  5=adobe pdf encryption |

### checkinRecord

Used to return a physical record (recordId) to a physical location (LocationRecordId) and the record state coding field is set to checkedin. The record state date is set to the date of the scan or the manual data entry. The coding fields site name, site number, aisle, bay and shelf are set using the LocationRecordId information. If the number of free space records at this LocationRecordId is greater than 0 then 1 free space record is deleted.

.

**POST:**

{webserver}/recordservice/checkinRecord/{login}/{password}/{recordId}/{locationRecordId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| recordId | unique record identifier |
| LocationRecordId | unique record location |

### checkoutRecord

Used to remove a physical record from a location and assign to a user or a location. You have to specify at least a user or a location or both. The record state coding field is set to checkedout and the record state date is set to the date of the scan or manual data entry. A free space is created at the LocationRecordId and all the coding filds are created.

**POST:**

{webserver}/recordservice/checkoutRecord/{login}/{password}/{RecordId}/{userRecordId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| RecordId | This is a unique record id of what you are checking out |
| UserRecordId | RecordId of the user you are checking out the item to |

### copyUserRecord

This function is used to copy a user from a user group to a functional group. The purpose of the function is to provide record security. The initial need for the function is for the publish program where you want to create a functional group for each customer so that they can see their records.

**POST:**

{webserver}/recordservice/copyUserRecord/{login}/{password}/{sourceObjectId}/{sourceObjectType}/{targetObjectId}/{trargetObjectType}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name = createService |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| sourceObjectId | This is the object id of the user you want to copy |
| sourceObjectType | This should be record type = user |
| targetObjectId | This is the object id of the parent node where you want to copy the user. This should be a storage object whose name is first letter of the last name |
| sourceObjectType | this should be record type = Storage |

### createDocumentRecord

This function is used to create a single eFile record type for a given organizationNumber under category, subject, topic and date. This will create the category, subject and topic structure if it does not already exist.

**POST:**

{webserver}/recordservice/createDocumentRecord/{login}/{password}/{category}/{subject}/{topic}/{date}/{recordName}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name = createService |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| category | This must exist |
| subject | This is optional |
| topic | this is optional |
| date | this is required and must have the format yyyy-mm-dd |

### createService

This restful interface is used to create a single service under Services storage container for a given organizationNumber.

**POST:**

{webserver}/recordservice/createService/{login}/{password}/{serviceName}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name = createService |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| serviceName | Name of the service to create |

### createRecordInDateBranch

This is used to create a record given the treeid of the parent container record. The record is stored by year, month and day. Use setRecordContentFields to set the record coding fields.

**POST:**

{webserver}/recordservice/createRecordInDateBranch/{login}/{password}/{containerTreeId}/{recordName}/{recordTypeFiller}/{branchDate}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name = createService |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| containerTreeId | This is an existing record type created in the admin interface example //fox/ocwd/1018 board/203/ Board of Director Files/203.01 Board Chron Packets/139 = 355689 |
| recordName | This is the name of the record in the directory tree |
| objectTypeRecord | This is the object type of the record you are adding for example a board chrom pact record = NRT191 |
| recordTypeFiller | Storage |
| branchDate | Supplied by the called to indicate date of record YYYY-MM-DD |

### deleteRecord

**Description:**

This restful interface is used to delete one directory record and all the children.

**POST:**

{webserver}/recordservice/deleteRecord/{login}/{password}/{objectId}/{objectType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Assigned login ID for access authentication |
| password | Password associated with login ID |
| objectId | Record id for what you want to delete |
| objectType | Record type for what you want to delete |

### deleteRecords

This call is used to delete multiple directory records and all the children of a given record.

**POST:**

{webserver}/recordservice/deleteRecords/{login}/{password}/

**CSV File:**

**each header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | RecordId | string | Unique record id for each item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |

### deleteRecordsByRecordTypeAndUser

This call is used to delete multiple directory records and all the children of a given record type for a given user.

**POST:**

{webserver}/recordservice/deleteRecordsByRecordTypeAndUser/{login}/{password}/{RecordType}/{UserRecordId}/

**CSV File:**

**each header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | RecordId | string | Unique record id for each item |
| 3 | RecordType | string | This filters the records by a single Record Type |
| 4 | UserRecordId | string | This is the secondary filter that filters the result by the user’s recordid. |

### emailLoginInfo

This restful interface is used to email login/password for a given device id.

**POST:**

{webserver}/recordservice/emailLoginInfo/{deviceId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| Logon | Unique string to identify user |
| deviceId | Unique Mobile device id |

### getCsvContent

This call is used to retrieve a csv file for a given directory id. There are many directory objects that have csv data attached and this call makes it easier to get the data for plotting, reporting and emailing to name a few operations.

**GET:**

{webserver}/recordservice/getCsvContent/{login}/{password}/{objectId}/{objectType}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| objectId | Record id |
| objectType | Record type |

### getLatLonByRecordType

This call gets all the coding fields for a given RecordType and filtered by coding fields. If the coding fields are blank then the system will return all latitude and longitude values for the given record type.

**GET:**

{webserver}/recordservice/getLatLonByRecordType/{login}/{password}/{RecordType}/{codingName1}/{codingValue1}/{codingName2}/{codingValue2}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Logon | Unique string to identify user |
| password | User’s password |
| RecordType | Selected record type example store |
| codingName1 | This is the coding field name for filtering example itemType |
| codingValue1 | This is the coding field value for filtering example customer |
| codingName2 | This is the coding field name for filtering example Region |
| codingValue2 | This is the coding field value for filtering example southwest |

Returns

NodeName

RecordId

Latitude

Longitude

### getLibraryCoding

This call gets all the coding fields for a given ItemNumber.

**GET:**

{webserver}/recordservice/getLibraryCoding/{login}/{password}/{ItemNumber}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| ItemNumber | This matches the ItemNumber in the accounting system |

### getLibraryItemsByRecordType

This call will return all library items of a given record type. We want to get all the items in the library that we don’t sync because they have a huge quantity like produce packs and we do this with transaction but we need to specify different record types when doing scanning in the field.

**GET:**

{webserver}/recordservice/getLibraryItemsByRecordType/{login}/{password}/{objectId}/{objectType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| RecordType |  |

### getRecordCoding

This restful interface is used to read the coding data for a particular record id (objectId, objectType).

**GET:**

{webserver}/recordservice/getRecordCoding/{login}/{password}/{objectId}/{objectType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| objectId | Internal Record id |
| objectType | Internal Record type |

### getRecordCodingByRecordId

This restful interface is used to read the coding data for a particular record id. This call returns the objectId, objectType and the coding fields.

**GET:**

{webserver}/recordservice/getRecordCodingByRecordId/{login}/{password}/{recordId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| recordId | This is recordId of the directory object you want |

### getRecordCodingByMobileRecordId

This restful interface is used to read the coding data for a particular mobile record id. This call returns the objectId, objectType and the coding fields.

**GET:**

{webserver}/recordservice/getRecordCodingByRecordId/{login}/{password}/{MobileRecordId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| MobileRecordId | This is mobileRecordId of the directory object you want |

### getRecordCount

This restful interface is used to get a count of the records of a given type.

**GET:**

{webserver}/recordservice/getRecordCount/login/password/itemType

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| itemType | Client, staff, vendor, part, service \*\*\*this is an enumerated type that can be a record except when you have a user and then you look at the coding |

### getRecordContent

This call is used to retrieve an image file for a given directory id. If you pass in a negative number for maxDimensionPixels it returns the bytes in the file otherwise it will scale the image.

**GET:**

{webserver}/recordservice/getRecordContent/{login}/{password}/{objectId}/{objectType}/{maxDimensionPixels}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| objectId | Record id |
| objectType | Record type |
| maxDimensionPixels | This is the desired image width in pixels (height is same) |

### getRecordContentAsByteData

This call is used to retrieve an image file for a given directory id. If you pass in a negative number for maxDimensionPixels it returns the bytes in the file otherwise it will scale the image. The reason for the filename argument is so that you can pass a complete file name and extension to the google docs viewer. The filename name has to mimic a file url through a web server. The reason for the byte stream is so that it is quicker than a json object.

**GET:**

{webserver}/recordservice/getRecordContent/{login}/{password}/{objectId}/{objectType}/{maxDimensionPixels}/{Filename}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| objectId | Record id |
| objectType | Record type |
| maxDimensionPixels | This is the desired image width in pixels (height is same) |
| Filename | This is the file name including the extension |

### getRecordContentUnencrypted

This restful interface is used to retrieve an eFile for a given directory id unencrypted. If the file is keyed then the system will use the key argument to decrypt. If encryption mode is auto then the system will use the password to decrypt.

**GET:**

{webserver}/recordservice/getRecordContent/{login}/{password}/{objectId}/{objectType}/{maxDimensionPixels}/{key}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| objectId | Record id |
| objectType | Record type |
| maxDimensionPixels | This is the desired image width in pixels (height is same) |
| key | This is used to decypt when the encryption mode is keyed |

### getRecordFilterCodingFields

This function returns an array of strings of all the Filter.XXXX for a given record type. This function is used by the mobile devices in setting up the arguments for the getNestedList function.

**GET:**

{webserver}/recordservice/getRecordFilters/{login}/{password}/{recordType}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| recordType | This is a record type for a specific record group (example Board Meeting Minutes for group electronic) |

### getRecordIdsAll

This restful interface is used to get ids and time stamps for a given itemType. Note that the itemType is mapped to the record type. The user record type needs the item type coding field and this is because several item types map to user but you only want the client item type and not the vendor or staff users.

**GET:**

{webserver}/recordservice/getRecordIdsAll/{login}/{password}/{itemType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| itemType | client, staff, vendor, part, service, store, category, subject, topic |

### getRecordIdsUpdated

This restful interface is used to get get ids and time stamps for a given displayTye under a directory node based on the maxTimestamp.

**GET:**

{webserver}/recordservice/getRecordIdsUpdated/{login}/{password}/{recordType}/{maxTimeStamp}

**EXAMPLE:**

{webserver}/recordservice/getRecordIdsUpdated/login/password//invoiceheader/0

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| password | User’s password for authentication |
| recordType | This is the record type coding field (eg. client, staff, vendor, part or service) |
| maxTimestamp | This is the maximum of content and content timestamps for all records of the given type |

### getRecordIdsAllByRecordType

This restful interface is used to get ids and time stamps for a given recordType.

**GET:**

{webserver}/recordservice/getRecordIdsAllbyRecordType/{login}/{password}/{recordType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| recordType | This is the rms record type name Eg. Timecard Detail |
| maxTimestamp | This is the maximum of content and content timestamps for all records of the given type |

### getRecordIdsUpdatedByRecordType

This restful interface is used to get ids and time stamps for a given recordType that have been updated since (greater than) the maxTimeStamp argument.

**GET:**

{webserver}/recordservice/getRecordIdsUpdatedbyRecordType/{login}/{password}/{recordType}/{maxTimeStamp}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| recordType | This is the rms record type name Eg. Timecard Detail |

### getRecordsUpdated

Used to get record data or ids depending on the parameter fullDataLimit the caller can handle. The recordType and maxTimestamp can be used to filter the data more although the call was designed for a sync process running in a low priority background thread on the mobile device. Note that when you get back the oject id you are getting and updating the coding fields for a particular object id.

**GET:**

{webserver}/recordservice/getRecordsUpdated/{login}/{password}/{recordType}/{fullDataLimit}/{maxTimestamp}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| password | User’s password for authentication |
| recordType | This is the rms record type name Eg. Timecard Detail |
| fullDataLimit | This is the number of records with full data (all coding fields) that the caller can handle. Example suppose you can handle 200 timecards (this could be header and detail records since the system may have been updated) and there are 500 timecards that meet the filter criteria. If more than the fullDataLimit then only 500 timecard header Id’s will be returned. |
| maxTimestamp | This is 0 is to get all records or a positive integer. |

### getRecordsUpdatedFiltered

Used to get record data or ids depending on the parameter fullDataLimit the caller can handle. The recordType and maxTimestamp can be used to filter the data more although the call was designed for a sync process running in a low priority background thread on the mobile device. Note that when you get back the oject id you are getting and updating the coding fields for a particular object id.

**GET:**

{webserver}/recordservice/getRecordsUpdatedFiltered/{login}/{password}/{recordType}/{fullDataLimit}/{maxTimestamp}/{filterCodingFieldName}/{filterCodingFieldValue}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| recordType | This is the rms record type name Eg. Timecard Detail |
| fullDataLimit | This is the number of records with full data (all coding fields) that the caller can handle. Example suppose you can handle 200 timecards (this could be header and detail records since the system may have been updated) and there are 500 timecards that meet the filter criteria. If more than the fullDataLimit then only 500 timecard header Id’s will be returned. |
| maxTimestamp | This is 0 is to get all records or a positive integer. |
| filterCodingFieldName | This is the coding field that will be used for filtering Example would be “Processed” |
| filterCodingFieldValue | This is the coding value that will be used for filtering Example would be “No” |

### getRecordsUpdatedX

Used to get record data or ids depending on the parameter fullDataLimit the caller can handle. The recordType and maxTimestamp can be used to filter the data more although the call was designed for a sync process running in a low priority background thread on the mobile device. Note that when you get back the oject id you are getting and updating the coding fields for a particular object id.

**GET:**

{webserver}/recordservice/getRecordsUpdatedX/{login}/{password}/{recordType}/{fullDataLimit}/{maxTimestamp}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| password | User’s password for authentication |
| recordType | This is the rms record type name Eg. Timecard Detail |
| maxNumberFullDataRecords | This is the number of records with full data (all coding fields) that the caller can handle. Example suppose you can handle 200 timecards (this could be header and detail records since the system may have been updated) and there are 500 timecards that meet the filter criteria. If more than the fullDataLimit then only 500 timecard header Id’s will be returned. |
| maxTimestamp | This is 0 is to get all records or a positive integer. |
| includeFields | Names of the coding fields you want separated by a comma |

### getRecordsUpdatedXFiltered

This is an enhanced **getRecordsUpdatedFilteredX.**

**GET:**

{webserver}/recordservice/getRecordsUpdatedXFiltered/{login}/{password}/{recordDisplayType}/{maxNumberfullDataRecords}/{maxTimestamp}/{fromDate}/{toDate}/{dateRangeField}/{filterFields}/{strFieldDelim}/{filterValues}/{strValueDelim}/{isFilterByFuncGroups}/{includeFields}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| password | User’s password for authentication |
| recordDisplayType | This is the rms record type name Eg. Timecard Detail |
| maxNumberFullDataRecords | This is the number of records with full data (all coding fields) that the caller can handle. Example suppose you can handle 200 timecards (this could be header and detail records since the system may have been updated) and there are 500 timecards that meet the filter criteria. If more than the fullDataLimit then only 500 timecard header Id’s will be returned. |
| maxTimestamp | This is 0 is to get all records or a positive integer. |
| fromDate | Start date of records you want |
| toDate | End date of records you want |
| dateRangeField | optional name of codingfield to apply date range filter to.  Required if fromDate or toDate provided. |
| filterFields | These are the coding field names you want to filter by |
| strFieldDelim | symbol used normally a , to separate filter fields |
| filterValues | This is the values used in the filter query |
| strValueDelim | symbol used normally a , to separate data fields |
| isFilterByFuncGroups | This is if you include function groups to filter data |
| includeFields | These are the coding fields you want returned |

I generalized  getRecordsUpdatedX in the back end (did not change the web API) as follows:

/\*\*  
  \*  
  \* @param user  
  \* @param securityCode  
  \* @param recordDisplayType  
  \* @param maxNumberFullDataRecords - if zero, gets all.  
  \* @param maxTimeStamp  
  \* @param fromDate - optional fromDate  
  \* @param toDate- optional toDate  
  \* @param dateRangeField - optional name of codingfield to apply date range filter to.  Required if fromDate or toDate provided.  
  \* @param filterFields - optional string holding deliminated list of codingfields to filter on.  
  \* @param strFieldDelim - optional delimiter for separating filter Fields. Defaults to comma.  
  \* @param filterValues - optional string holding deliminated list of codingfield values to filter on.  Must match the list of filter fields.  
  \* @param strValueDelim - optional delimiter for separating filter values. Defaults to comma.  
  \* @param isFilterByFuncGroups - true if you want to filter by user's functional groups.  
  \* @param includeFields - optional string holding a comma separated list of codingfield names you want returned.  If empty, get all.  
  \* @return  
  \* @throws Exception  
  \*/  
public Object[] getRecordsUpdatedX(  
  User user,  
  String securityCode,  
  String recordDisplayType,  
  String maxNumberFullDataRecords,  
  String maxTimeStamp,  
  String dateRangeField,  
  String fromDate,  
  String toDate,  
  String filterFields,  
  String strFieldDelim,  
  String filterValues,  
  String strValueDelim,  
  String isFilterByFuncGroups,  
  String includeFields) throws Exception

This allows us to develop a very flexible **getRecordsUpdatedFilteredX**.

### getRecordInfo

This is used to get all the part ids for a given store number and a given item id.

**GET:**

{webserver}/recordservice/getRecordInfo/{login}/{password}/{coding name}/{coding value}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique string to identify user |
| password | User’s password |
| Coding name | This is the coding name of the record to use for the search |
| Coding value | This is a unique value |

### getRecordTypesByGroup

This returns a string array of all the different record types for a given group.

**GET:**

{webserver}/recordservice/getRecordTypes/{login}/{password}/{coding name}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique string to identify user |
| password | User’s password |
| groupName | This is the group {electronic, organization, physical, people} |

### getTemplatePartIds

This restful interface is used to get all the part ids for a given store number and a given item id.

**GET:**

{webserver}/recordservice/getTemplatePartIds/{login}/{password}/{item id}

**EXAMPLE:**

{webserver}/recordservice/getTemplatePartIds/login/password//602/500-100-0001/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| storeNumber | This is the given store number from which you want all the part ids. |
| Item id | This is the item id of the parent object |

### movePhysicalRecord

This is used to move a record in the directory to a new location. LocationRecordId will be updated. The free space record counters in the old and new container record will be decremented and incremented.

**POST:**

{webserver}/recordservice/movePhysicalRecord/{login}/{password}/{recordId}/{locationRecordId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| recordId | unique record identifier |
| LocationRecordId | This is the new record location id of where the record will go |

### setRecordsByRecordTypeAndUser

This call creates record data for setup and/or testing.

**GET:**

{webserver}/shipservice/setRecordsByRecordTypeAndUser/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | organizationName | string | This is the organization name |
| 8 | organizationNumber | string | This is the organization number |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Returns:**

Number of locations created/modified/deleted

Number+error message string

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setRecordCodingTimestamp

This restful interface is used to set the specified records coding timestamp.

**GET:**

{webserver}/directoryservice/setRecordCodingTimestamp/{login}/{password}/{timeStamp}/{objectId}/{objectType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| timestamp | Internal timestamp to indicate one or more coding fields has been changes for the given record. |
| objectId | Record id |
| objectType | Record type |

### setRecordContent

This call updates the efile content for a given directory id. Normally this is a photo but could be a PDF file or even a url address.

**POST:**

{webserver}/recordservice/setRecordContent/{login}/{password}/{objectId}/{objectType}{key}{Mode}

**EXAMPLE:**

curl -k -X POST -F [media=@6441.txt](mailto:media=@6441.txt) {webserver}/recordservice/setRecordContent/login/password/6440/User Note that 6441.txt is the file to be uploaded to the server.

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| objectId | object id |
| objectType | object type |
| Key | Encryption key for server. If blank then no encryption else aes256 |
| Mode | Encryption mode for server.  0=no encryption  1=use password for encryption  2=use supplied key string  5=adobe pdf encryption |
| RecordId |  |

### setRecordContentFile

This function updates the efile content for multiple records.

**POST:**

{webserver}/recordservice/setRecordContentFile/{login}/{password}/

CSV File

objectId, objectType, filename CRLF, path to file, delete

\*\*\*\* If you do not put in the word delete then the file is left on the path.

\*\*\*\* You must first copy the filename to the path on the server

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | ation code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |

### setRecordCoding

This call updates a single coding value for a given record.

**POST:**

{webserver}/recordservice/setRecordCoding/{login}/{password}/{objectId}/{objectType}/{codingName}/{codingValue}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| objectId | Record id |
| objectType | Record type |
| codingName | This is the coding name text field |
| codingValue | This is the value associated with a given codingName |

### setRecordCodingByRecordId

This call updates a given record using the recordid.

**POST:**

{webserver}/recordservice/setRecordCodingByRecordId/{login}/{password}/

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | Not needed |
| 4 | objectType | string | Not needed |
| 5 | MobileRecordId | String | Not needed |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | organizationName | string | This is the organization name |
| 8 | organizationNumber | string | This is the organization number |
| 9 | RecordId | String |  |
| 10 | UpdateTimestamp | String |  |
| 11 | Coding Field Name | String |  |
| 12 | Coding Field Value | string |  |

**Notes**

1. ObjectId & ObjectType

"Object ID","231","Object Type","NRT309","Status","In Transit","SetCodingTimestamp","1482492548774"

2.RecordId

"Object ID","839639","Object Type","[[recordid]]","Status","In Transit","SetCodingTimestamp","1482492548774"

POST call

<https://www.rcofox.com/Image2000/rest/recordservice/setRecordCodingFields/henry/henry>

### setRecordContentTimestamp

This restful interface is used to set the specified record content timestamp.

**POST:**

{webserver}/recordservice/setRecordContentTimestamp/{login}/{password}/{timeStamp}/{displayType}/{objectId}/{objectType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | setRecordContentMaxTimestamp - action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| objectId |  |
| objectType | Record type |
| Timestamp | This is the content timestamp |
| displayType |  |

### setRecordName

This restful interface is used to set the record name (record directory title).

**POST:**

{webserver}/recordservice/setRecordName/{login}/{password}/{objectId}/{objectType}/{newName}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| objectId | Record id |
| objectType | Record type |
| newName | This is the record name you want to assign to this directory id |

### setRecordCodingFields

This restful interface is used to update or set multiple coding fields for a directory. This is a post method with file attachment that contains coding field name and value pairs separated by a CRLF delimiter. The first 2 values of each line are the objectId and objectType of the record you want to modify its coding fields.

**POST:**

{webserver}/recordservice/setRecordCodingFields/{login}/{password}/

Example of Curl command:

curl -k -X POST -F media=@fields.txt [{webserver}/](https://www.rcofox.com/Image2000/rest/)

recordservice/setRecordCodingFields/login/password//1013/NRT182

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | objectId | string | objectId of the record you want to set its coding fields |
| 2 | ObjectType | string |  |
| 3 | Coding Field1 Name | String | The name is in quotes Example “PTN” |
| 4 | Coding Field1 Value | string | The value is in quotes Example “1234” |
| 5 | Coding FieldN Name | String |  |
| 6 | Coding FieldN Value | String |  |

I think the correct format is one record per line, i.e. no linefeeds between codingfields, with a comma separating codingfields (instead of a linefeed).

So the correct CSV file data should look something like this (all one line, may look wrapped):

"LobjectId","9909","objectType","User","Telephone","","MobileRecordId","","Organization Name","CMT","Device ID","","RMS User Id","9902","ItemType","client","FunctionalGroupName","Public","Address2","","Address1","AAA Full Transportation","Item Discount","","Latitude","","Company","AAA Full Transportation, 55","RecordId","80861","Organization Number","9","Service Discount","","Country","","ZipCode","11234","Title","--","City","Tempe","Email","[info@AAAFullTransportation.com](mailto:info@AAAFullTransportation.com)","State","AZ","Home URL","","Longitude","","Last Name","AAA Full Transportation","First Name","55"

**\*\*\*Note**

You may be updating multiple records in your csv file. You always have the objectid and objectidValue at the start of the line following by all the coding field names and values on each line. You then put a carriage return at the end of each line.

Example

You have 3 different directory records you are changing.

“LobjectId”,”3456”,”First Name”,”Mary”

“LobjectId”,”6656”,”First Name”,”Jane”

“LobjectId”,”7756”,”First Name”,”Charm”

### setServiceRequests

This creates service requests separated by yyyy/mm/dd (receive date). The directory name is Request Company. If Request Company is blank then Requestor Last Name, Requestor First Name.

**POST:**

{webserver}/recordservice/setServiceRequests/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | string | This is used by the device |
| 6 | Functional Group Name | string | This is security for login user what they can see |
| 7 | Request Id | string | From dbtext to identify records |
| 8 | Request Date | date |  |
| 9 | Due Date | date |  |
| 10 | Finish Date | date |  |
| 11 | Request Company | string |  |
| 12 | Request Last Name | string |  |
| 13 | Request First Name | string |  |
| 14 | Address1 | string |  |
| 15 | City | string |  |
| 16 | State | string |  |
| 17 | Zip | string |  |
| 18 | Email | string |  |
| 19 | Received By | string |  |
| 20 | Assigned To | string |  |
| 21 | Description | string |  |
| 22 | Notes | string |  |
| 23 | Date Received | Date |  |
| 24 | Completed Date | Date |  |

## Report Service

The following calls are database generated data for reports and or charts.

### getInvoiceChartData

This will return the cumulative invoice sum for the give login for each month and show a solid line for known sums and a dashed line for the remaining months. The function will return an array of the following. If you leave start date and end date blank then you get the current. The date format is YYYY-MM-DD and you need 2 digits so zero pad.

["Year","Month","Cumulative Sum",”Flag”]

["Year","Month","Cumulative Sum",”Flag”]

["Year","Month","Cumulative Sum",”Flag”]

…

Flag = E for Estimated and A for Actual data

**GET:**

{webserver}/reportservice/getInvoiceChartData/{login}/{password}/{StartDate}/{NumberYears}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| StartDate | Start date with default = 2015-01-01 |
| NumberYears | Integer number years default = 1 |

### getTimecardChartData

This will return the cumulative timecard total hours sum for the give login for each month and show a solid line for known sums and a dashed line for the remaining months. The function will return an array of the following. If you leave start date and end date blank then you get the current. The date format is YYYY-MM-DD and you need 2 digits so zero pad.

["Year","Month","Cumulative Sum",”Flag”]

["Year","Month","Cumulative Sum",”Flag”]

["Year","Month","Cumulative Sum",”Flag”]

…

Flag = E for Estimated and A for Actual data

**GET:**

{webserver}/reportservice/getTimecardChartData/{login}/{password}/{StartDate}/{NumberYears}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| StartDate | Start date default = 2015-01-01 |
| NumberYears | The number of year with default = 1 |

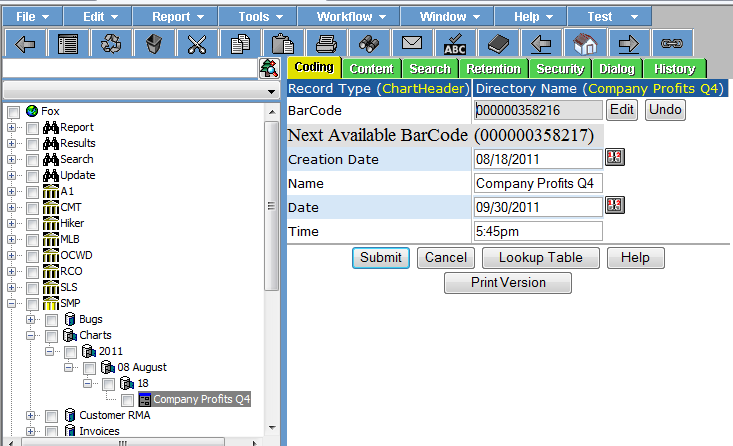
## Quickbook Service

The following web services are used by the quick books application to communicate with the rms server.

### createChartHeader

This service creates an eFile record type under the Chart storage directory. When you create the chart you give it a name specified by the user. Note if the name already exists for the given date you replace the content of the exisiting name.

The Print2Pdf driver calls the web service with a name, date and time and the rms set the coding fields with arguments and also sets the directory node name with the name argument. The following figure shows a sample chart header node. The node is an electronic record type.



**POST:**

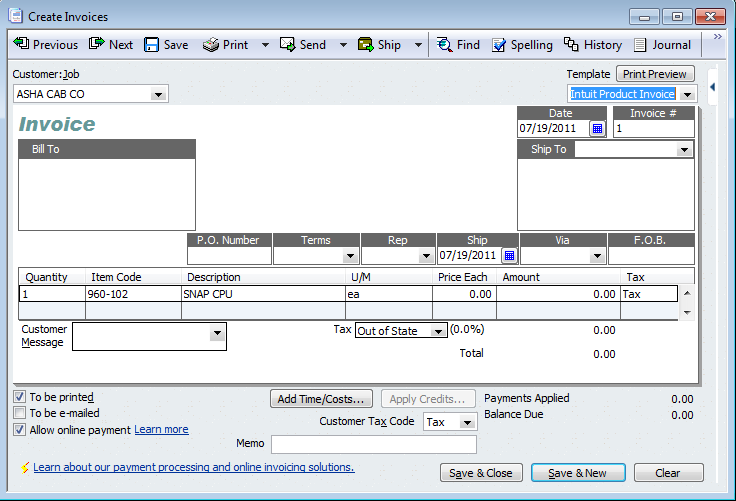
{webserver}/quickbookservice/createChartHeader/{login}/{password}/{chartName}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getRecordMaxTimestamp - action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| chartName | User supplied chart name when they printed |

### createInvoiceHeader

This creates an invoice header record. The quick book scheduler run the synchronize program that monitors quick books invoice and add them to the rms.

****

**POST:**

File: {webserver}/quickbookservice/createInvoiceHeader/{login}/{password}/

**File**

H, Invoice Number, Mobile Invoice Number, Device Id, Customer:Job, Date, Purchase Order, terms, rep, Ship Date, Via, F.O.B., Ship To Addreess, Ship To CityStateZip, Ship To Name, Bill To Address, Bill To CityStateZip, Bill To Name, Tracking Number, Processed, Total, IsSale, Location, Item Type, Year, Month, Day, Organization Name, Organization Number CRLF

### createInvoiceDetails

This creates an invoice detail record under the Item Receipt Header Record. There can be one or more detail records per header.

**POST:**

File,{webserver}/quickbookservice/createInvoiceDetails/{login}/{password}/

**File**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. | |
| 2 | Invoice Number | string | This is from the accounting system invoice | |
| 5 | MobileRecordId | String | This is mobile invoice number | |
| 6 | Device ID | string |  | |
| 7 | Quantity | string |  | |
| 8 | Item Number | string |  | |
| 9 | Description | string |  | |
| 10 | U/M | string |  | |
| 11 | Price (ea) | string |  | |
| 12 | Amount | string |  | |
| 13 | Tax | String |  | |
| 14 | Quantity Shipped | String |  | |
| 15 | Ship To Name | String |  | |
| 16 | Customer:Job | String |  | |
| 17 | CustomerRecordId | String |  | |
| 18 | Quantity Not Picked | String |  | |
| 19 | Quantity Not Shipped | String |  | |
| 20 | Item Type | String |  | |
| 21 | Organization Name | String |  | |
| 22 | Organization Number | String |  | |
| 23 | ObjectId | String |  | |
| 24 | ObjectType | String |  | |

### createItemReceiptHeader

This creates an Item Receipt header record. When the mobile operator receives new inventory you want to generator what a quick books shipping clerk would do manually by clicking the Receive Inventory in the home screen. This function creates a header record that has one or more detail records.

****

The accounting synchronization method sets the header Processed coding field to yes after all detail records have been processed. When the item receipt header is first created the processed field is set to no. Setting the field to no makes the searches for all new item receipts that have not been processed run faster.

**POST:**

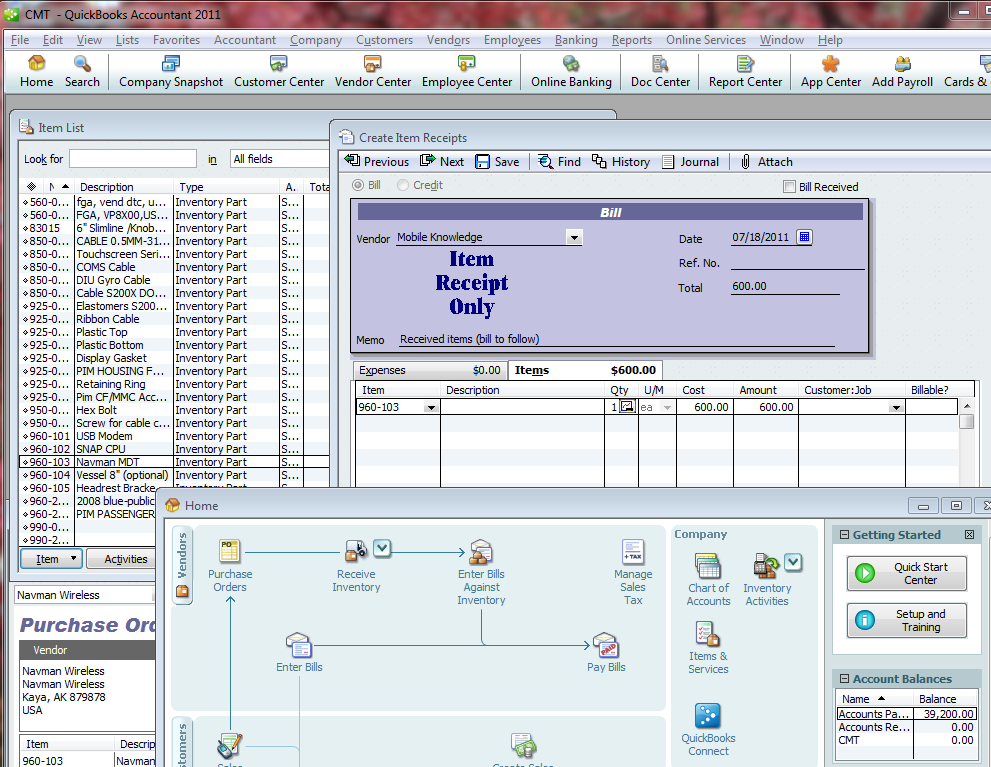
{webserver}/quickbookservice/createItemReceiptHeader/{login}/{password}/{vendor name}/{purchase order number}/{date}{Reference Number}/{total}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| vendorName | The mobile operator |
| Purchase order Number | The mobile operator scans this number on the package |
| Date | This is the date of item was scanned in receving |
| Reference Number | This is for locating the item receipt. Could the vendors control number for the shipped item. |
| Total | This is the sum total for all items |

### createItemReceiptDetails

This creates an Item Receipt detail record under the Item Receipt Header Record. There can be one or more detail records per header item receipt.

****

**POST:**

{webserver}/quickbookservice/createItemReceiptDetails/{login}/{password}/{parentObjectId}/{parentObjectType}{item number}/{item description}/{vendor name}/{customer}/{quantity}/{unitOfMeasure}/{cost}/{amount}/{billable}/{receiptDate}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| parentObjectId | This is the id of the item receipt parent |
| parentObjectType | This is the object type which should be Item Receipt Header |
| Item number | This is the item id |
| Item description | In quick book this is Description on Sales Transaction |
| Vendor name |  |
| quantity |  |
| u/m |  |
| cost |  |
| Amount |  |
| billable |  |

### createPayment

This creates a payment record under the Payments storage.

**POST:**

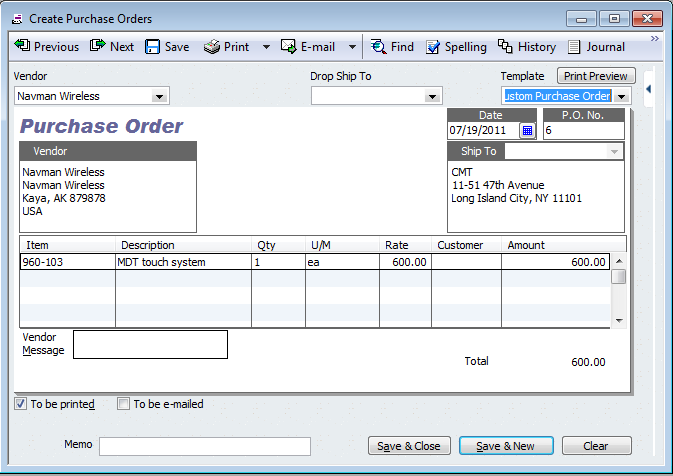
{webserver}/quickbookservice/createPayment/{login}/{password}/{name}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Name | This is the directory name of the payment record |

### createPurchaseOrderHeader

This creates a Purchase Order Header record. We want to keep quickbooks synchronized with the rms at all times.



**POST:**

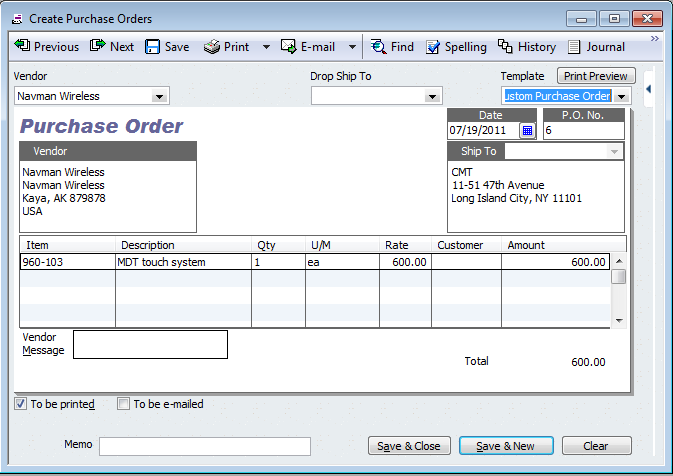
{webserver}/quickbookservice/createPurchaseOrderHeader/{login}/{password}/{vendor name}/{Drop Ship To}/{purchase order number}/{date}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| vendorName | The mobile operator |
| Purchase order Number | The mobile operator scans this number on the package |
| Date | This is the purchase order date |
| Drop ship to | This is where the purchase order items will delivered |

### createPurchaseOrderDetails

This creates a detail record under the Purchase Order Header Record. There can be one or more detail records per header.



**POST:**

{webserver}/quickbookservice/createPurchaseOrderDetails/{login}/{password}/{parentObjectId}/{parentObjectType}/{purchaseOrderNumber}/{itemId}/{itemDescription}/vendorName}/{quality}/{unitsOfMeasure}/{rate}/amount}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| parentObjectId | This is the id of the item receipt parent |
| parentObjectType | This is the object type which should be Item Receipt Header |
| purchaseOrderNumber | Accounting synchronize will add purchase order to each line item for reporting purposes |
| itemId | This is the part item number |
| itemDescription |  |
| vendorName | Name of the vendor you are buys parts from |
| quantity | Quantity of items you are buying from the selected vendor |
| unitsOfMeasure | Units of measure for the item |
| rate | This is the cost the vendor is charging you for the item |
| amount | Total amount = rate \* quantity for this line item |

### createRefund

This creates a refund record under the Refunds storage.

**POST:**

{webserver}/quickbookservice/createRefund/{login}/{password}/{name}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Name | This is the directory name of the payment record |

### createRMADetail

This creates the rma detail record for a given record. The detail will correspond with each line in the invoice.

**POST:**

{webserver}/quickbookservice/createRMADetail/{loginId}/{password}/ …. (see arguments in the below table)

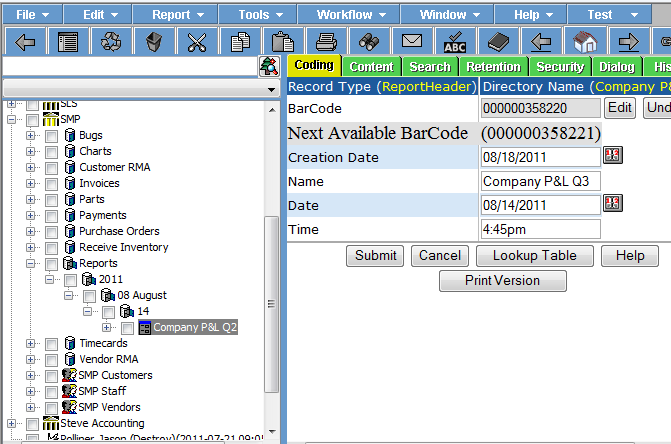
**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| parentObjectId | This is the timecard parent id |
| parentObjectType | This is the timecard parent record type |
| invoiceNumber |  |
| invoiceDate |  |
| rmaNumber |  |
| clientName |  |
| clientNumber |  |
| itemName |  |
| itemNumber |  |
| partId |  |
| rmaValue |  |
| description |  |
| quantityToReturn |  |
| quantityShipped |  |
| quantityReturned |  |

### createReportHeader

This service creates an eFile record type under the Chart storage directory. When you create the chart you give it a name specified by the user. Note if the name already exists for the given date you replace the content of the exisiting name.

The Print2Pdf driver calls the web service with a name, date and time and the rms set the coding fields with arguments and also sets the directory node name with the name argument.



**POST:**

{webserver}/quickbookservice/createReportHeader/{login}/{password}/{name}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getRecordMaxTimestamp - action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| reportName | User supplied report name when they printed the report |

### createRMAHeader

This creates the RMA parent record with the data obtained from an invoice number.

**POST:**

{webserver}/quickbookservice/createRmaHeader/{login}/{password}/{invoice number}/{invoice date}/{ rmaNumber}/{ clientName}/{ clientNumber }

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| invoiceNumber | This is the invoice where the client wants to return one or more parts |
| invoiceDate | This is the invoice date when created and sent to client |
| rmaNumber |  |
| clientName |  |
| clientNumber | This uniquely identifies a client |

### createVoid

This creates a void record under the Voids storage.

**POST:**

{webserver}/quickbookservice/createVoid/{login}/{password}/{name}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Name | This is the directory name of the payment record |

### getAccounts

This call gets all the accounts and their associated coding fields from the Rms Coding Timestamp

**GET:**

{webserver}/quickbookservice/getAccounts/{login}/{password}/{maxTimeStamp}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| maxTimestamp | This is the maximum of content and content timestamps for all records of the given type. If not passed returns all records |

### getBills

This function gets all bills (header+details+coding fields) where invoice header for the userRecordId and for CodingName = Processed and CodingValue=no. You can only get bills where the login is a member of the functional group.

**GET:**

{webserver}/quickbookservice/getInvoices/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |

### getCustomerRmaDetailIds

This is used to read the customer rma detial ids for a given rma and organization.

**GET:**

{webserver}/quickbookservice/getCustomerRmaDetailIds/{login}/{password}/{parentObjectId}/{parentObjectType}/

**EXAMPLE:**

{webserver}/quickbookservice/getCustomerRmaDetailIds/login/password//11/NRT202/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Parent object id | This id is obtained from the getRmaHeaderId |
| Parent object type | This id is obtained from the getRmaHeaderId |

### getCustomerRmaHeaderId

This is used to get the customer rma header id for a given customer rma number for a given organization.

**GET:**

{webserver}/quickbookservice/getCustomerRMAHeaderId/{login}/{password}/{CustomerRmaNumber}

**EXAMPLE:**

{webserver}/quickbookservice/getCustomerRMAHeaderId/login/password//9

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| CustomerRmaNumber | This is the number assigned by the factory and sent to the customer so they can return material |

### getCustomerRMAs

This call returns all customers header and detail coding information for processed flag with given date range.

**GET:**

{webserver}/quickbookservice/getCustomerRMAs/{login}/{password}/{CustomerRecordId}/{Processed}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| CustomerRecordId | This is the customer you want the rma’s for |
| Processed | If yes give the closed rms’s else no = open rma’s |

### getItemReceiptDetailIds

This function gets the item receipt details ids for a given item receipt header id.

**GET:**

{webserver}/quickbookservice/getItemReceiptDetailIds/{login}/{password}/{parentObjectId}/{parentObjectType}/

**EXAMPLE:**

{webserver}/quickbookservice/getItemReceiptDetailIds/login/password//2/NRT204

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| parentObjectId | This is the record id of the item receipt parent record |
| parentObjectType | This is the record type of the item receipt parent record which should be itemReceiptHeader |

### getChartHeaderIdsUpdated

This function gets all chart header ids where the content timestamp has changed.

**GET:**

{webserver}/quickbookservice/getChartHeaderIdsUpdated/{login}/{password}/{maxTimeStamp}/

**EXAMPLE:**

{webserver}/quickbookservice/getChartHeaderIdsUpdated/login/password//0/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |

### getChartHeaderIdsAll

This function gets all chart header ids

**GET:**

{webserver}/quickbookservice/getChartHeaderIdsAll/{login}/{password}/

**EXAMPLE:**

{webserver}/quickbookservice/getChartHeaderIdsAll/login/password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |

### getContracts

This call gets all the open contracts and their associated coding fields that the login has access too.

**GET:**

{webserver}/quickbookservice/getContracts/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |

### getDealers

This function gets all dealers (header+coding fields).

**GET:**

{webserver}/quickbookservice/getInvoices/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |

### getExpenses

This call gets all the expenses and their associated coding fields from the Rms Coding Timestamp

**GET:**

{webserver}/quickbookservice/getExpenses/{login}/{password}/{maxTimeStamp}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| maxTimeStamp | 0=all or value=new than the given time stamp |

### getInvoices

This function gets all invoices (header+details+coding fields) where invoice header for the userRecordId and for CodingName = Processed and CodingValue=no.

**GET:**

{webserver}/quickbookservice/getInvoices/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |

### getInvoicesByDateAndSalesRep

This function gets all invoices Date and Invoice Total from Invoice Header for the give Sales Rep (this is their userRecordId). This function will return 2 arrarys (Date, Total). This function is used in the timecard chart to plot the cumulative sales for a give rep.

**GET:**

{webserver}/quickbookservice/getInvoicesByDateAndSalesRep/{login}/{password}/{startDate}/{EndDate}

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| startDate | Default is start of current year |
| endDate | This is current date |

### getInvoiceHeaderId

This is used to read the invoice hearder record ids for a given invoice number for a particular user.

**GET:**

{webserver}/quickbookservice/getInvoiceHeaderId/{login}/{password}/{invoiceOrderNumber}

EXAMPLE:  
{webserver}/quickbookservice/getInvoiceHeaderId/login/password//3

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| invoiceNumber | This is the accounting invoice number |

### getInvoiceDetailIds

This function gets the invoice details ids of the invoice header record types for a specific date range.

**GET:**

{webserver}/quickbookservice/getInvoiceDetailIds/{login}/{password}/{parentObjectId}/{parentObjectType}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| parentObjectId | This is the record id of the item receipt parent record |
| parentObjectType | This is the record type of the item receipt parent record which should be itemReceiptHeader |

### getMatters

This function gets the matters the login has access to.

**GET:**

{webserver}/quickbookservice/getMatters/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| practiceAreaNumberFilter | Filter matter by practice area |
| startDateFilter | Filter mattes by date range |
| endDateFilter | Filter mattes by date range |
| clientNumnberFilter | Filter matters by client number |

### getNewInvoiceHeaderIds

This function gets the new invoice header ids where the processed coding field equals no.

**GET:**

{webserver}/quickbookservice/getNewInvoiceHeaderIds/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| parentObjectId | This is the record id of the item receipt parent record |
| parentObjectType | This is the record type of the item receipt parent record which should be itemReceiptHeader |

### getNewItemReceiptHeaderIds

This function gets the new item receipt details ids for a given item receipt header id. All ids with the coding field Process = no get returned.

**GET:**

{webserver}/quickbookservice/getNewItemReceiptHeaderIds/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |

### getNewPaymentHeaderIds

This function gets the new payment header ids where the processed coding field = no. A payment record can not be set to yes until the full amount has been received.

**POST:**

{webserver}/quickbookservice/getNewPaymentHeaderIds/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |

### getOpenInvoices

This call returns a list of invoice header ids that have not been completely processed. The invoice header coding field Processed can have multiple states {partial, complete, open}

**GET:**

{webserver}/quickbookservice/getOpenInvoices/{login}/{password}/{invoiceOrderNumber}

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| invoiceNumber | This is the accounting invoice number |

### getOpenPurchaseOrders

This call returns a list of purchase order header ids that have not been completely processed. The invoice header coding field Processed can have multiple states {partial, complete, open}

**GET:**

{webserver}/quickbookservice/getOpenPurchaseOrders/{login}/{password}/{purchaseOrderNumber}

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| purchaseOrderNumber | This is the accounting purchase order number |

### getPayment

This call returns a payment record information given a transaction id.

**GET:**

{webserver}/quickbookservice/getPayment/{login}/{password}/{transaction id}

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| Transaction Id | This is the transaction id of the payment received from processor |

### getPaymentByMobileInvoiceNumber

This call returns a payment record and the coding fields given a mobile invoice number.

**GET:**

{webserver}/quickbookservice/getPaymentByMobileInvoiceNumber/{login}/{password}/{mobileInvoiceNumber}

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| mobileInvoiceNumber | This number is generated by the mobile device and emailed or printed and sent/given to customer |

### getPickList

This call returns a list of locations (aisle, bay, shelf) for a given invoice number.

**GET:**

{webserver}/quickbookservice/getPickList/{login}/{password}/{invoiceOrderNumber}/{factoryStoreNumber}

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| invoiceOrderNumber | This is the accounting invoice number you want the locations sorted on the server. |
| factoryStoreNumber | This is the store where the picker is getting the items from |

### getPurchaseOrderDetailIds

This function gets the purchase order details ids of the purchase order header record types for a specific date range.

**POST:**

{webserver}/quickbookservice/getPurchaseOrderDetailIds/{login}/{password}/{parentObjectId}/{parentObjectType}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| parentObjectId | This is the record id of the item receipt parent record |
| parentObjectType | This is the record type of the item receipt parent record which should be itemReceiptHeader |

### getPurchaseOrderHeaderId

This is used to read the purchase order hearder record ids for a given purchase order for a particular user.

**GET:**

{webserver}/quickbookservice/getPurchaseOrderHeaderId/{login}/{password}/{purchaseOrderNumber}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| PurchaseOrderNumber |  |
|  |  |

### getPurchaseOrders

This function gets all purchase orders (header+details+coding fields) where invoice header coding field Processed=no.

**GET:**

{webserver}/quickbookservice/getPurchaseOrders/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |

### getQuantityOnHand

This function is used to get a total count for each part by reading all the stores parts’ quanity on hand values and updating the quickbooks inventory parts’ quantity on hand. There are a few important coding fields for a part record type that are important in creating the desired lists as shown in the following table.

|  |  |
| --- | --- |
| **Coding Field** | **Description** |
| Item Name / Number | This is the part identifier |
| Quantity On Hand | This coding field indicates how many parts are “believed” to be in the inventory stores. |
| Organization Number | This is used for security so that you only process codes for the user’s organization |
| Store Number | This is used to indicate what store a part is located in. Note that a part must have a parent container of record type store and every part has a store number. |
| Deployment | This is either manufacturing or field to let you group the location of where the parts are located. |

The resulting output list contains Item Name / Number, Quantity On Hand.

**GET:**

{webserver}/quickbookservice/getQuantityOnHand/{login}/{password}/{Filter}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |

### getRefund

This call returns a refund record information given a transaction id.

**GET:**

{webserver}/quickbookservice/getRefund/{login}/{password}/{transaction id}

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| Transaction Id | This is the transaction id of the payment received from processor |

### getRefundDetailIds

This function gets the invoice details ids of the refund header record types for a specific date range.

**GET:**

{webserver}/quickbookservice/getRefundDetailIds/{login}/{password}/{parentObjectId}/{parentObjectType}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| parentObjectId | This is the record id of the item receipt parent record |
| parentObjectType | This is the record type of the item receipt parent record which should be itemReceiptHeader |

### getRefunds

This call returns all refunds including header and detail coding information for a given mobile invoice number or invoice number.

**GET:**

{webserver}/quickbookservice/getRefunds/{login}/{password}/{mobileInvoiceNumber}/{InvoiceNumber}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| mobileInvoiceNumber | This is the invoice number generated by a mobile device |
| InvoiceNumber | This is the invoice number generated by the accounting software and a synchronization has occurred between the accounting software and the RMS. |

### getRefundsByTimestamp

This call returns all refunds including header and detail coding information greater than or equal to a timestamp.

**GET:**

{webserver}/quickbookservice/getRefunds/{login}/{password}/{timestamp}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| Timestamp | Return all invoices for an organization greater than or equal to the given timestamp |

### getReportHeaderIdsAll

This function gets all report header ids.

**GET:**

{webserver}/quickbookservice/getReportHeaderIdsAll/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |

### getReportHeaderIdsUpdated

This function gets all report header ids where the content timestamp has changed.

**GET:**

{webserver}/quickbookservice/getReportHeaderIdsUpdated/{login}/{password}/{maxTimeStamp}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| maxTimeStamp | This is 0 to get all time stamps or the last maxTimeStamp you have stored in your local database tables |

### getSalesOrders

This function gets all sales orders (header+details+coding fields) where invoice header for the userRecordId and for CodingName = Processed and CodingValue=no.

**GET:**

{webserver}/quickbookservice/getSalesOrders/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |

### getSalesTaxCodes

This call gets all the sales tax codes and their associated coding fields from the MaxTimestamp

**GET:**

{webserver}/quickbookservice/getSalesTaxCodes/{login}/{password}/{MaxTimeStamp}/

**CSV File:**

salesTaxCode,salesTaxCodeInactive,description,taxable,itemType,organizationName,organizationNumber CRLF

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| maxTimestamp | This is the maximum of content and content timestamps for all records of the given type |

### getSalesTaxRates

This call gets all the sales tax rates and their associated coding fields from the MaxTimeStamp.

**GET:**

{webserver}/quickbookservice/getSalesTaxRates/{login}/{password}/{MaxTimeStamp}/

**CSV File:**

salesTaxId, salesTaxRate,city,state,itemType,organizationName,organizationNumber CRLF

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| maxTimestamp | This is the maximum of content and content timestamps for all records of the given type |

### getServices

This call gets all the services and their associated coding fields from the Rms Coding Timestamp

**GET:**

{webserver}/quickbookservice/getServices/{login}/{password}/ maxTimeStamp}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| maxTimestamp | This is the maximum of content and content timestamps for all records of the given type. If not passed returns all records |

### getTerms

This call gets all the terms and their associated coding fields from the Rms Coding Timestamp

**GET:**

{webserver}/quickbookservice/getTerms/{login}/{password}/ {maxTimeStamp}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| maxTimestamp | This is the maximum of content and content timestamps for all records of the given type. If not passed returns all records |

### getTransactions

This call gets transactions filtered by type which is optional, login, and timestamp. The limit specifies the maximum number of records where you want all the data for a record returned. If the number of transaction is greater than the limit then only the objectId and objectType are returned.

**GET:**

{webserver}/quickbookservice/getTerms/{login}/{password}/{ItemType}/{Processed}/{limit}/{ maxTimeStamp}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| ItemType | If this is blank then return payments, refunds and voids otherwise only return transactions that match the ItemType |
| Processed | If blank then all transactions otherwise return yes or no processed transactions. |
| Limit | This specifies the maximum number of records that should be returned with all the objectId, objectType and all the coding data. |
| maxTimestamp | This is the maximum of content and content timestamps for all records of the given type. If not passed returns all records |

### getVendorRmaDetailIds

This is used to read the vendor rma detail ids for a given rma and organization.

**GET:**

{webserver}/quickbookservice/getVendorRmaDetailIds/{login}/{password}/{parentObjectId}/{parentObjectType}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| Parent object id | This id is obtained from the getRmaHeaderId |
| Parent object type | This id is obtained from the getRmaHeaderId |

### getVendorRmaHeaderId

This is used to get the vendor rma header for an vendor rma number and organization.

**GET:**

{webserver}/quickbookservice/getVendorRMAHeaderId/{login}/{password}/{VendorRmaNumber}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| VendorRmaNumber | This is the rma number assigned by the vendor to return the material. |

### getVendorRMAs

This function gets the vendor rma header and detail for processed coding field and date range.

**GET:**

{webserver}/shipservice/getVendorRMAs/{login}/{password}/}/{VendorRecordId}{Processed}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| VendorRecordId | This is the vendor you are sending items to for whatever reason |
| Processed | Coding field is yes or no |

### getVoid

This call returns a void record information given a transaction id.

**GET:**

{webserver}/quickbookservice/getVoid/{login}/{password}/{transaction id}

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| Transaction Id | This is the transaction id of the payment received from processor |

### incrementInvoiceDetailQtyPicked

This increments the coding field quantity Picked in the invoice detail record each time the picker process X items off the invoice order.

**GET:**

{webserver}/quickbookservice/createRmaHeader/{login}/{password}/{invoiceNumber}/{ItemNunber}/{RecordId}/{quantity}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| invoiceNumber | This is the invoice number currently being processed by picker |
| RecordId | This is the record id of the item getting picked |
| Quantity | This is the number that just got picked |

### setAccounts

This function creates/updates accounts. The attached csv file has the following format. The web services will check if the account coding field called “Number” exists and update coding fields otherwise a new account is created and the coding fields are set. The directory name is formed by using Number Name.

**POST:**

{webserver}/quickbookservice/setAccounts/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. | |
| 2 | Flag | string | “H” - Indicates this is a invoice header item | |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. | |
| 4 | objectType | string | If invoice header exists then this is Invoice Header | |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp | |
| 6 | FunctionalGroupName | string | This is the group responsible for the record | |
| 7 | number | string | Accounting chart of accounts number | |
| 8 | name | string | Chart of accounts name | |
| 9 | description | string | Description of what the does | |
| 10 | subitemOf | string | Used if account is a subitem of another account | |
| 11 | active | string | Is the account active | |
| 12 | Account Type | string | Is the account expense, income | |
| 13 | orgName | String | Name of the organization | |
| 14 | orgNumber | String | Number of the organization | |

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setBills

This function creates/updates bills. The attached csv file uses a CRLF character to separate records. The csv file must contain at least one header line and zero or more detail lines. Each header line must start with an H character and each detail line must start with the D character and must contain the Invoice Number to link the detail and header record.

The current rule for setBills(), etc, is that if a CSV row has non-blank ObjectId and ObjectType values, it the corresponding record will be updated.  If those values are blank, a new record will be created in the database.

There is a special case to point out:  if the mobile device wants to add a new Detail record under an existing Header record, in the CSV file you must include a row for the existing Header record with ObjectId and ObjectType, even if you do not need to update it -- then you can follow with any number of rows for new detail records.

When quickbooks generates an invoice the mobileCustomerNumber, mobileBillToNumber and mobileShipToNumber are set using the objectId of the respective customers.

In the rms directory the names of the nodes are as follows.

For the header (Vendor Name – Bill Number)

For the detail (account,  Expense Amount – item, Item Amount)

**POST:**

{webserver}/quickbookservice/setBills/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | MobileBillNumber | string | This number is created by the mobile device and is formed by combining the mobile unique device id-internal auto incrementing invoice |
| 8 | deviceId | string | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 9 | Vendor Name | string | Name of the vendor from bill drop down list |
| 10 | Vendor RecordId | string | Number of the vendor |
| 11 | Address1 | string | This is the address of where items will get delivered. |
| 12 | City | string | City |
| 13 | State | string | State |
| 14 | ZipCode | string | Zip code |
| 15 | Terms | string | These are the payment terms supplied by mobile device or quickbooks invoice form |
| 16 | Memo | string | This is the notes section of the bill |
| 17 | Date | Date | This is the creation date of the bill. |
| 18 | Reference Number | string | This is usually something from vendor bill |
| 10 | Amount Due | Fractional # | Bill total amount sum(expenses) + sum(items) |
| 20 | Due Date | Date | Date the bill is due |
| 21 | itemType | string | This is billheader |
| 22 | organizationName | string | This is the organization name to whom the customer belongs |
| 23 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 24 | processed | string | The quickbooks service sets this rms coding value to processed after a synchronization |
| 25 | Transaction State | string | This can be paid, unpaid or other |
| 26 | Transaction Status | string | This can be paid, unpaid, other |
| 27 | Expenses Total | Fractional # | This is the sum of all the expenses |
| 28 | Items Total | Fractional # | Total all the items |
| 29 | Bill Number | String | Assigned by rms |
| 30 | ExpenseName | String | Assigned by mobile user |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” =this is an invoice detail item |
| 3 | objectId | string | This is the objectId for the detail if you are updating a record otherwise it is blank. |
| 4 | objectType | string | This is the objectType for the detail if you are updating a record otherwise it is blank |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | Header Bill Number | Whole Number | From header |
| 8 | mobileBillNumber | string | This number is created by the mobile device and is formed by combining the mobile unique device id-internal auto incrementing invoice |
| 9 | deviceId | string | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 10 | Account | String | From chart of accounts |
| 11 | Amount | Fractional # | Amount of expense or item |
| 12 | Memo | string | This describes the expense |
| 13 | Customer:Job | string | This is the customer that will receive items and is selected from the customer list on mobile device |
| 14 | Contract Number | string | There may be several contracts per customer:job |
| 15 | Contract Name | string | There may be several contracts per customer:job |
| 16 | Billable | Boolean | Is expense or item detail chareable to customer |
| 17 | Item Number | string | This is from the quickbooks item list |
| 18 | Description | string | This is the description |
| 19 | Quantity | Fractional # | This is the amount of tax on the item |
| 20 | Cost | Fractional # | This is the person who will receive items |
| 21 | itemType | string | This is BillExpenseDetail or BillItemDetail You must keep expense and Item details on separate lines!!! |
| 22 | Class | string | This is the bill to customer record id |
| 23 | Organization Name | string | This is the organization name to whom the customer belongs |
| 24 | Organization Number | string | This is the organization number to whom the customer belongs. |
| 25 | ExpenseName | String | Assigned by mobile user |

Each data line is separated by a CRLF. Data fields within a line are separated by a comma. All data elements are surrounded by quotes. You can have a header record and no details if you are doing an update on the header. To update a detail you must have the header record.

If the Invoice Numbrer or Mobile Invoice Number exist then you are doing an update operation otherwise you are doing a record creation.

Example of Curl command:

curl -k -X POST -F [media=@fields.txt](mailto:media=@fields.txt) [{webserver}/](https://www.rcofox.com/Image2000/rest/)quickbookservice/setInvoices/login/password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setDealers

This function creates quickbooks pro advisor. The attached csv file has the following format. The directory name is the Company name.

**POST:**

{webserver}/quickbookservice/setDealers/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. | |
| 2 | Flag | string | “H” - Indicates this is a invoice header item | |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. | |
| 4 | objectType | string | If invoice header exists then this is Invoice Header | |
| 5 | Company | String | company name if none then last name, first name | |
| 6 | First Name | string | First name | |
| 7 | Last Name | string | Last name | |
| 8 | Address1 | string | Full address | |
| 9 | City | string | City | |
| 10 | State | string | State | |
| 11 | ZipCode | string | Zip code can contain – or space and the final 4 zipcode | |
| 12 | Phone Number | string | Telephone number | |
| 13 | Mobile Phone | String | Mobile phone number | |
| 14 | Email | String | Email address | |
| 15 | Notes1 | String | Sales persons notes | |
| 16 | Description1 | String | Services offered | |

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setExpenses

This function creates/updates expenses. The attached csv file has the following format. The web services will check if the expense coding field called “Number” exists and update coding fields otherwise a new expense is created and the coding fields are set. The directory name is formed by using Number Name.

**POST:**

{webserver}/quickbookservice/setExpenses/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | FunctionalGroupName | string | This is the group responsible for the record |
| 6 | Number | string | Expense account number from chart of accounts |
| 7 | Name | string | Expense account name from chart of accounts |
| 8 | Description | string | Description of the expense |
| 9 | Tax Line | string | This is tax line deductions |
| 10 | orgName | String | Name of the organization |
| 11 | orgNumber | String | Number of the organization |

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setInvoices

This function creates/updates invoices. The attached csv file uses a CRLF character to separate records. The csv file must contain at least one header line and zero or more detail lines. Each header line must start with an H character and each detail line must start with the D character and must contain the Invoice Number to link the detail and header record.

The current rule for setInvoices(), etc, is that if a CSV row has non-blank ObjectId and ObjectType values, it the corresponding record will be updated.  If those values are blank, a new record will be created in the database.

There is a special case to point out:  if the mobile device wants to add a new Detail record under an existing Header record, in the CSV file you must include a row for the existing Header record with ObjectId and ObjectType, even if you do not need to update it -- then you can follow with any number of rows for new detail records.

The precedence rule that the back end code uses is that it will first look for an object id, objecttype and then it will look for a mobile recordid and then an invoice # and finally a mobile invoice #. If all those fail the system will not update the record.

1. objectid objecttype
2. mobilerecordid
3. invoice #
4. mobile invoice #

When Intuit Quickbooks generates an invoice the mobileCustomerNumber, mobileBillToNumber and mobileShipToNumber are set using the objectId of the respective customers.

When Microsoft Dynamics AX creates an invoice it will set the invoice # to the AX invoice number and the mobile invoice # will be ax concatenated with the invoice #.

When Sage 100 or Sage 200 creates an invoice it will set the invoice # to the Sage invoice number and the mobile invoice # will be sage concatenated with the invoice #.

**POST:**

{webserver}/quickbookservice/setInvoices/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | organizationName | string | This is the organization name to whom the customer belongs |
| 8 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 9 | invoiceNumber | string | This number is set by the quickbooks service and is a unique positive number created by quickbooks |
| 10 | mobileInvoiceNumber | string | This number is created by the mobile device and is formed by combining the mobile unique device id-internal auto incrementing invoice |
| 11 | deviceId | string | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 12 | customerJob | string | This unique string identifies each quickbooks customer or job. |
| 13 | CustomerRecordId | string | This is the bill to customer record id |
| 14 | Date | Date | This is the creation date of the invoice. |
| 15 | purchaseOrder | string | This is the customer supplied purchase order |
| 16 | Terms | string | These are the payment terms supplied by mobile device or quickbooks invoice form |
| 17 | Rep | string | This is the sales representative filled out on mobile device |
| 18 | shipDate | Date | This is the date the invoice items were shipped and is filled out by shipping department using mobile def |
| 19 | Via | string | This is the name of the shipping carrier |
| 20 | Fob | string | This is to indicate freight on board for taxes |
| 21 | Ship To Company | string | This is the person who will receive invoice items |
| 22 | Ship To Name | string | This is the person who will receive invoice items |
| 23 | Ship To Address | string | This is the person who will receive invoice items |
| 24 | Ship To City | String |  |
| 25 | Ship To State | String |  |
| 26 | Ship To Country | String |  |
| 27 | Ship To Zipcode | String |  |
| 28 | Ship To CityStateZip | String |  |
| 29 | Bill To Company | string | This is the address of where items will get paid. |
| 30 | Bill To Name | string | This is the address of where items will get paid. |
| 31 | Bill To Address | string | This is the address of where items will get paid. |
| 32 | Bill To City | string | This is the address of where items will get paid. |
| 33 | Bill To State | string | This is the address of where items will get paid. |
| 34 | Bill To Country | string | This is the address of where items will get paid. |
| 35 | Bill To Zipcode | string | This is the address of where items will get paid. |
| 36 | billToCityStateZip | string | This is the city, state and zip information of where the items will get paid from. |
| 37 | trackingNumber | string | This number is supplied by the shipping carrier to track the invoice |
| 38 | Processed | string | The quickbooks service sets this rms coding value to processed after a synchronization |
| 39 | Total | Fractional # | This is the invoice total amount |
| 40 | isSale | string | If yes then this invoice contains one or more items where the amount is > 0 |
| 41 | Location | string | This is a location id of where the item is going to |
| 42 | itemType | string | This is invoiceheader |
| 43 | mobileCustomerNumber | string | This is used by the mobile device to identify a customer. This is the objectId of customer. |
| 44 | mobileBillToNumber | string | This is used by the mobile device to the bill to customer. This is the objectId of customer. |
| 45 | mobileShipToNumber | string | This is used by the mobile device to identify the ship to customer. This is the objectId of customer. |
| 46 | Store Name | string | This is the name of the store where the items come from |
| 47 | Store Number | string | This is the store number where the items come from. |
| 48 | Notes | string | This is the notes section of the invoice |
| 49 | TransactionState | string | This can be paid, unpaid or refund |
| 50 | TransactionStatus | string | This can be paid, unpaid, refund, unknown |
| 51 | Payment Method | string | check, cash or credit card |
| 52 | Transaction Id | string | This is the id supplied by processor if payment method = credit card |
| 53 | Check Number | string | This is the check number if payment method = check |
| 54 | Reference Number | string | This is the same as check number and is used by quick books |
| 55 | Sales Tax | string | This is the total sales tax of all detail items |
| 56 | Sales Tax Rate | string | This is the sales tax rate for the given invoice |
| 57 | Currency | string | This is the currency for the details |
| 58 | Number of Packages | Integer | Number of packages in shipment |
| 59 | Freight Amount | String |  |
| 60 | Shipping Weight | String |  |
| 61 | Ship Zone | String |  |
| 62 | Carrier | String |  |
| 63 | Ship Status | String |  |
| 64 | CreatorRecordId | String |  |
| 65 | Processed Date | String |  |
| 66 | Reseller Code | String |  |
| 67 | Process Name | String |  |
| 68 | LocationRecordId | string |  |
| 69 | BillOfLadingRecordId | string |  |

**Detail Line terminated with CRLF**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |  |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. | |
| 2 | Flag | string | “D” =this is an invoice detail item |  |
| 3 | objectId | string | This is the objectId for the detail if you are updating a record otherwise it is blank. |  |
| 4 | objectType | string | This is the objectType for the detail if you are updating a record otherwise it is blank |  |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp | |
| 6 | FunctionalGroupName | string | This is the group responsible for the record | |
| 7 | organizationName | string | This is the organization name to whom the customer belongs |  |
| 8 | organizationNumber | string | This is the organization number to whom the customer belongs. |  |
| 9 | invoiceNumber | string | This number is set by the quickbooks service and is a unique positive number created by quickbooks |  |
| 10 | mobileInvoiceNumber | string | This number is created by the mobile device and is formed by combining the mobile unique device id-internal auto incrementing invoice |  |
| 11 | deviceId | string | Mobile device unique device id obtained from the rms when mobile device first initialized |  |
| 12 | Quantity | Fractional # | This is the quantity ordered |  |
| 13 | itemNumber | string | This is the quickbooks item list number |  |
| 14 | Description | string | This describes the item |  |
| 15 | Unit of Measure | string | Unit of measure normally blank |  |
| 16 | Price Each | Fractional # | This is the unit cost of the item |  |
| 17 | Amount | Fractional # | Quantity \* priceEach |  |
| 18 | Tax | String | This is the amount of tax on the item |  |
| 19 | Ship To Name | string | This is the person who will receive items |  |
| 20 | Customer:Job | string | This is the customer that will receive items and is selected from the customer list on mobile device |  |
| 21 | CustomerRecordId | string | This is the bill to customer record id |  |
| 22 | quantityPicked | Fractional # | This is the number of items picked from a store to fill the order and is filled out by shipping department. |  |
| 23 | quantityNotPicked | Fractional # | This is the number of items not picked from a store to fill the order and is filled out by shipping department. |  |
| 24 | quantityShipped | Fractional # | This is the actual quantity shipped filled out by shipping department |  |
| 25 | QuantityNotShipped | Fractional # | This is the number of items not shipped to customer and is filled out by shipping department |  |
| 26 | Quantity Delivered | Fractional # | This is the number of items not shipped to customer and is filled out by shipping department |  |
| 27 | itemType | string | This should be part |  |
| 28 | Store Name | string | This is the name of the store where the items come from |  |
| 29 | Store Number | string | This is the store number where the items come from. |  |
| 30 | Processed | Fractional # | Freight amount for this shipment |  |
| 31 | Notes | Fractional # | Freight amount for this shipment |  |
| 32 | Date | Fractional # | Freight amount for this shipment |  |
| 33 | Freight Amount | Fractional # | Freight amount for this shipment |  |
| 34 | Ship Zone | string | Where the shipment is going |  |
| 35 | Shipping Weight | Fractional # | Total weight of shipment |  |
| 36 | Number of Packages | Integer | Number of packages in shipment |  |
| 37 | Price Code | String | Used for discounts |  |
| 38 | Pallet Serial Number | String | Used for truck loading |  |
| 39 | Items | String | Used for truck loading |  |

Each data line is separated by a CRLF. Data fields within a line are separated by a comma. All data elements are surrounded by quotes. You can have a header record and no details if you are doing an update on the header. To update a detail you must have the header record.

If the Invoice Numbrer or Mobile Invoice Number exist then you are doing an update operation otherwise you are doing a record creation.

Example of Curl command:

curl -k -X POST -F [media=@fields.txt](mailto:media=@fields.txt) [{webserver}/](https://www.rcofox.com/Image2000/rest/)quickbookservice/setInvoices/login/password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setMatters

This function creates/updates matter. The attached csv file uses a CRLF character to separate records. The csv file must contain at least one header line and zero or more detail lines. Each header line must start with an H character.

**POST:**

{webserver}/quickbookservice/setMatters/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Argument** | | **Data Type** | **Description** |
| 1 | | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | | string | “H”-Indicates this is a header item |
| 3 | objectId | | string | This is only for an update |
| 4 | objectType | | string | This is only for an update |
| 5 | | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | Organization Name | | string | Org name |
| 8 | Organization Number | | string | Org number |
| 9 | Customer Name | | string | Company Name or Last Name, First Name |
| 10 | CustomerRecordId | | string | This is the transaction id generated by the processor |
| 11 | PracticeAreaNumber | | string | Legal practice area |
| 12 | PracticeAreaName | | String | This describes the practice |
| 13 | MatterNumber | | string | Number of the matter |
| 14 | MatterDescription | | string | Describes the matter |
| 15 | Date | | string | Date the matter started |

### setPayments

This function creates/updates payments. The attached csv file uses a CRLF character to separate records. The csv file must contain at least one header line and zero or more detail lines. Each header line must start with an H character.

**POST:**

{webserver}/quickbookservice/setPayments/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Argument** | | **Data Type** | **Description** |
| 1 | | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | | string | “H”-Indicates this is a header item |
| 3 | objectId | | string | This is only for an update |
| 4 | objectType | | string | This is only for an update |
| 5 | | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | Customer Name | | string | First and last name of the customer |
| 7 | Transaction Id | | string | This is the transaction id generated by the processor |
| 8 | Invoice Number | | string | This is the accounting invoice number if available |
| 9 | Mobile Invoice Number | | string | This is the mobile invoice number |
| 10 | Total Amount | | Number | This is the amount of the transaction |
| 11 | Payment Method | | string | How is the payment bein made {Cash, check, credit card} |
| 12 | Payment Amount | | string | This is the amount of the payment and must match the invoice total amount |
| 13 | Check Number | | string | Check number if payment method equals check |
| 14 | Processed | | Boolean | True or False |
| 15 | Date | | string | Date the payment transaction was sent |
| 16 | Time | | string | Time the payment transaction was sent |
| 17 | | FunctionalGroupName | string | This is the group responsible for the record |
| 18 | OrganizationName | | string | Name of the vendor’s organization |
| 19 | OrganizationNumber | | string | Number of the vendor’s organization |
| 20 | Transaction Cancelled | | Boolean | True or False |
| 21 | Void Processed | | string | Not used now |
| 22 | DeviceId | | string | This is the mobile device id |
| 23 | ItemType | | string | payment |
| 24 | Customer Record Id | | string | This is the customer record id |
| 25 | Memo | | string | Notes about the transaction |
| 26 | Reference Number | | string | This is the customer reference number if available |
| 27 | Originator | | string | ?? |

### setPurchaseOrders

This function creates/updates purchase orders. The attached csv file uses a CRLF character to separate records. The csv file must contain at least one header line and zero or more detail lines. Each header line must start with an H character and each detail line must start with the D character and must contain the Purchase Order Number to link the detail and header record.

**POST:**

{webserver}/quickbookservice/setPurchaseOrders/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | string | “H”-Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | FunctionalGroupName | string | This is the group responsible for the record |
| 6 | organizationName | string | This is the organization name to whom the customer belongs |
| 7 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 8 | purchaseOrder | string | This number is set by the quickbooks service and is a unique positive number created by quickbooks |
| 9 | Date | Date | Date of the purchase order/ customer rma/ vendor return |
| 10 | VendorName | string |  |
| 11 | VendorRecordId | string |  |
| 12 | Drop ship to | string |  |
| 13 | Processed | string |  |
| 14 | ProcessedDate | string |  |
| 15 | ReceiveStoreName | string | Name of the selected customer or vendor |
| 16 | ReceiveStoreRecordId | string | This record id of the vendor or customer store |
| 17 | ReceivedFromStoreName | string | This record id of the vendor or customer store |
| 18 | ReceivedFromStoreRecordId | string | Name of the selected customer or vendor |
| 19 | DestinationStoreName | string |  |
| 20 | DestinationStoreRecordId | string |  |
| 21 | Tax | string |  |
| 22 | Tax Code | string |  |
| 23 | Freight | string |  |
| 24 | Notes | String |  |
| 25 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 26 | mobilePurchaseOrder | string | This number is created by the mobile device and is formed by combining the mobile unique device id-internal auto incrementing invoice |
| 27 | Device Id | string | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 28 | itemType | string | This is PurchaseOrderHeader |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | string | “D”-Indicates this is a invoice detail item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | organizationName | string | This is the organization name to whom the customer belongs |
| 8 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 9 | Purchase Order | string | This number is set by the quickbooks service and is a unique positive number created by quickbooks |
| 10 | mobilePurchaseOrder | string | This number is created by the mobile device and is formed by combining the mobile unique device id-internal auto incrementing invoice |
| 11 | Item Number | string | This is the item number of what you are buying |
| 12 | Description | string | This describes the item number |
| 13 | Unit of Measure | string | This is the unit of measure for item number |
| 14 | Quantity | string | Number of item number ordered |
| 15 | Quantity Received | string | Number of item number ordered |
| 16 | Quantity Last Received | string | Number of item number ordered |
| 17 | Cost | string | This is the unit price of item number |
| 18 | Amount | string | This is the total amount of the order |
| 19 | Billable | string | Is this item for resale or internal |
| 20 | Date | string | This is the date order was submitted |
| 21 | Customer:Job | string | This is who the items are for if resale |
| 22 | LocationRecordId | string | This is who the items are for if resale |
| 23 | Processed | String | yes or no |
| 24 | Processed Date | Date | When the po detail was processed |
| 25 | Notes | String | Notes about the detail |
| 26 | Device Id | string | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 27 | Item Type | String | This is always “purchaseorderdetail” |

Each data line is separated by a CRLF. Data fields within a line are separated by a comma. All data elements are surrounded by quotes. You can have a header record and no details if you are doing an update on the header. To update a detail you must have the header record.

If the Invoice Numbrer or Mobile Invoice Number exist then you are doing an update operation otherwise you are doing a record creation.

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setPurchaseOrders |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

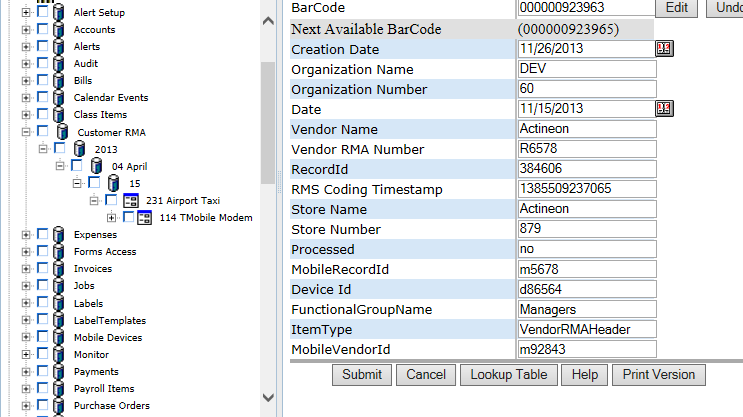
### setCustomerRMAs

This function creates/updates receive inventory header and detail. The attached csv file uses a CRLF character to separate records. The csv file must contain at least one header line and zero or more detail lines. Each header line must start with an H character and each detail line must start with the D character and must contain the Invoice Number to link the detail and header record.

The current rule is that if a CSV row has non-blank ObjectId and ObjectType values, it the corresponding record will be updated.  If those values are blank, a new record will be created in the database.

There is a special case to point out:  if the mobile device wants to add a new Detail record under an existing Header record, in the CSV file you must include a row for the existing Header record with ObjectId and ObjectType, even if you do not need to update it -- then you can follow with any number of rows for new detail records.

The directory is formed by adding nodes under the Customer RMA branch followed by year, month, day branches. Then you have your cusotmer rma header and custmer rma detail. The cusotmer rma header consists of the cusotmer rma number (auto generated by the rms server) followed by a space character and then the cusotmer name. For the cusotmer rma detail the directory node name is composed of the item number followed by a space character and then the description.



**POST:**

{webserver}/quickbookservice/setCustomerRMAs/{login}/{password}/

Example of Curl command:

curl -k -X POST -F [media=@fields.txt](mailto:media=@fields.txt) [{webserver}/](https://www.rcofox.com/Image2000/rest/)quickbookservice/setInvoices/login/password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | Device Id | string | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 8 | Date | Date | This is the creation date of the customer RMA. |
| 9 | Processed | string | The quickbooks service sets this rms coding value to processed after a synchronization |
| 10 | Customer RMA Number | String | This is the factory rma if items come from customer |
| 11 | ItemType | string | This is CustomerRMAHeader |
| 12 | Organization Name | string | This is the organization name to whom the customer belongs |
| 13 | Organization Number | string | This is the organization number to whom the customer belongs. |
| 14 | MobileCustomerId | string | This is used by the mobile device to identify a customer. This is the objectId of customer. |
| 15 | Store Name | string | This is the name of the store where the items come from |
| 16 | Store Number | string | This is the store number where the items come from. |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | String | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. | |
| 2 | Flag | String | “D” =this is an invoice detail item |
| 3 | objectId | String | This is the objectId for the detail if you are updating a record otherwise it is blank. |
| 4 | objectType | String | This is the objectType for the detail if you are updating a record otherwise it is blank |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp | |
| 6 | FunctionalGroupName | String | This is the group responsible for the record | |
| 7 | DeviceId | String | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 8 | Quantity to Return | Fractional # | This is the quantity to Return |
| 9 | Quantity Returned | Fractional # | Total number of items received to date |
| 10 | Quantity Received | Fractional # | The quantity just received |
| 11 | Item Number Received | String | This is the quickbooks item list number |
| 12 | Item Number Shipped | String | This is the quickbooks item list number |
| 13 | Description Item Received | String | This describes the item |
| 14 | Description Item Shipped | String | This describes the item |
| 15 | Customer RMA Number | String | This is the factory rma number given to the customer and is auto-generated by the system |
| 16 | ItemType | String | This should be CustomerRMADetail |
| 17 | organizationName | String | This is the organization name to whom the customer belongs |
| 18 | organizationNumber | String | This is the organization number to whom the customer belongs. |
| 19 | Store Name | String | This is the name of the store where the items come from |
| 20 | Store Number | String | This is the store number where the items come from. |
| 21 | Notes | String | If return why item is returned |
| 22 | Processed | String | This shows the detail is finished |
| 23 | Quantity Shipped | String | This is how many we have shipped |
| 24 | Quantity Not Shipped | String | This is how many we still have to ship |

Each data line is separated by a CRLF. Data fields within a line are separated by a comma. All data elements are surrounded by quotes. You can have a header record and no details if you are doing an update on the header. To update a detail you must have the header record.

If the Invoice Numbrer or Mobile Invoice Number exist then you are doing an update operation otherwise you are doing a record creation.

### setCustomerRMAsNoReturnData

This is the same call as setCustomerRMAs except only a single message is returned. The purpose of the call is for importing existing data into the RMS for a conversion

**POST:**

{webserver}/quickbookservice/setCustomerRMAsNoReturnData/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | Device Id | string | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 8 | Date | Date | This is the creation date of the customer RMA. |
| 9 | Processed | string | The quickbooks service sets this rms coding value to processed after a synchronization |
| 10 | Customer RMA Number | String | This is the factory rma if items come from customer |
| 11 | ItemType | string | This is CustomerRMAHeader |
| 12 | Organization Name | string | This is the organization name to whom the customer belongs |
| 13 | Organization Number | string | This is the organization number to whom the customer belongs. |
| 14 | MobileCustomerId | string | This is used by the mobile device to identify a customer. This is the objectId of customer. |
| 15 | Store Name | string | This is the name of the store where the items come from |
| 16 | Store Number | string | This is the store number where the items come from. |

**Detail Line terminated with CRLF**

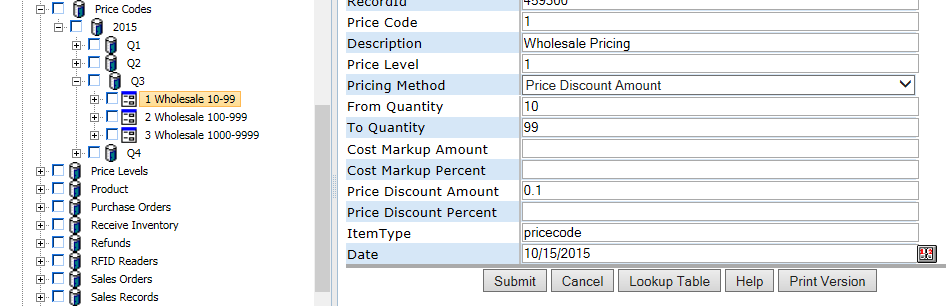
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | String | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. | |
| 2 | Flag | String | “D” =this is an invoice detail item |
| 3 | objectId | String | This is the objectId for the detail if you are updating a record otherwise it is blank. |
| 4 | objectType | String | This is the objectType for the detail if you are updating a record otherwise it is blank |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp | |
| 6 | FunctionalGroupName | String | This is the group responsible for the record | |
| 7 | DeviceId | String | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 8 | Quantity to Return | Fractional # | This is the quantity to Return |
| 9 | Quantity Returned | Fractional # | Total number of items received to date |
| 10 | Quantity Received | Fractional # | The quantity just received |
| 11 | itemNumber | String | This is the quickbooks item list number |
| 12 | Description | String | This describes the item |
| 13 | Customer RMA Number | String | This is the factory rma number given to the customer and is auto-generated by the system |
| 14 | ItemType | String | This should be CustomerRMADetail |
| 15 | organizationName | String | This is the organization name to whom the customer belongs |
| 16 | organizationNumber | String | This is the organization number to whom the customer belongs. |
| 17 | Store Name | String | This is the name of the store where the items come from |
| 18 | Store Number | String | This is the store number where the items come from. |
| 19 | Notes | String | If return why item is returned |
| 20 | Processed | String | This shows the detail is finished |
| 21 | Quantity Shipped | String | This is how many we have shipped |
| 22 | Quantity Not Shipped | String | This is how many we still have to ship |

Each data line is separated by a CRLF. Data fields within a line are separated by a comma. All data elements are surrounded by quotes. You can have a header record and no details if you are doing an update on the header. To update a detail you must have the header record.

If the Invoice Numbrer or Mobile Invoice Number exist then you are doing an update operation otherwise you are doing a record creation.

### setPriceCodes

This call creates/updates a list of price codes. The list is stored under a node called price codes and is a simple list with price code description as the directory node name. The settor will use the date to store the price code by year and quarter. Note that price codes should be created uniquely by the accounting system so you can trace an invoice or sales order containing price codes.



**POST:**

{webserver}/quickbookservice/setPriceCodes/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | Device Id | string | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 8 | Organization Name | string | This is the organization name to whom the customer belongs |
| 9 | Organization Number | string | This is the organization number to whom the customer belongs. |
| 10 | MobileCustomerId | string | This is used by the mobile device to identify a customer. This is the objectId of customer. |
| 11 | Price Code | string | This is the price code identifier |
| 12 | Price Code Description | string | Describes the price code |
| 13 | Pricing Method | List | Cost markup amount, cost markup percent, price discount amount, price discount percent |
| 14 | Pricing Value | Float |  |
| 15 | From Quantity | string |  |
| 16 | To Quantity | string |  |
| 17 | ItemType | string | pricecode |
| 18 | Effective Date | Date | Start of the special |
| 19 | End Date | Date | End of the special |
| 20 | Price Level Code | String |  |
| 21 | Price Level Description | String |  |

### setPriceLevels

This call creates/updates a list of price levels for a customer. The list is stored under a node called price levels and is a simple list with price level description as the directory node name.

**POST:**

{webserver}/quickbookservice/setPriceLevel/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | Device Id | string | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 8 | Organization Name | string | This is the organization name to whom the customer belongs |
| 9 | Organization Number | string | This is the organization number to whom the customer belongs. |
| 10 | Price Level Code | String | Price level identifier |
| 11 | Price Level Description | String | Describes the Price Level |
| 12 | Item Type | String |  |
| 13 | CustomerRecordId | string | Identifies the customer |
| 14 | Customer Name | string | Name of the customer this is company |
| 15 | Item Number | string | Part or service |
| 16 | Item Number Description | String | This is the item number description from the library |
| 17 | Effective Date | Date | Start date when price level enabled |
| 18 | End Date | Date | End date when price level disabled |

### setRecordProcessed

I also created **CQuickbooksServices.setRecordProcessed(String login, String passwd,  
  String objectId, String objectType)** .

Don can make a service for it and the mobile unit can call it it after each header is processed to set the codingfield "Processed" to "yes".

### setRefunds

This function creates/updates refunds. The attached csv file uses a CRLF character to separate records. The csv file must contain at least one header line and zero or more detail lines. Each header line must start with an H character and each detail line must start with the D character and must contain the Invoice Number to link the detail and header record.

The current rule for setRefunds(), etc, is that if a CSV row has non-blank ObjectId and ObjectType values, it the corresponding record will be updated.  If those values are blank, a new record will be created in the database.

There is a special case to point out:  if the mobile device wants to add a new Detail record under an existing Header record, in the CSV file you must include a row for the existing Header record with ObjectId and ObjectType, even if you do not need to update it -- then you can follow with any number of rows for new detail records.

When quickbooks generates an invoice the mobileCustomerNumber, mobileBillToNumber and mobileShipToNumber are set using the objectId of the respective customers.

**POST:**

{webserver}/quickbookservice/setInvoices/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H”=this is an invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | invoiceNumber | string | This number is set by the quickbooks service and is a unique positive number created by quickbooks |
| 8 | mobileInvoiceNumber | string | This number is created by the mobile device and is formed by combining the mobile unique device id-internal auto incrementing invoice |
| 9 | deviceId | string | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 10 | customerJob | string | This unique string identifies each quickbooks customer or job. |
| 11 | Date | Date | This is the creation date of the invoice. |
| 12 | purchaseOrder | string | This is the customer supplied purchase order |
| 13 | Terms | string | These are the payment terms supplied by mobile device or quickbooks invoice form |
| 14 | Rep | string | This is the sales representative filled out on mobile device |
| 15 | shipDate | Date | This is the date the invoice items were shipped and is filled out by shipping department using mobile def |
| 16 | Via | string | This is the name of the shipping carrier |
| 17 | Fob | string | This is to indicate freight on board for taxes |
| 18 | shipToAddress | string | This is the address of where items will get delivered. |
| 19 | shipToCityStateZip | string | This is the city, state and zip information of where the items will get delivered by shipping carrier |
| 20 | shipToName | string | This is the person who will receive invoice items |
| 21 | billToAddress | string | This is the address of where items will get paid. |
| 22 | billToCityStateZip | string | This is the city, state and zip information of where the items will get paid from. |
| 23 | billToName | string | This is the name of the person paying the invoice bill |
| 24 | trackingNumber | string | This number is supplied by the shipping carrier to track the invoice |
| 25 | processed | string | The quickbooks service sets this rms coding value to processed after a synchronization |
| 26 | Total | Fractional # | This is the invoice total amount |
| 27 | isSale | string | If yes then this invoice contains one or more items where the amount is > 0 |
| 28 | location | string | This is a location id of where the item is going to |
| 29 | itemType | string | This is invoiceheader |
| 30 | organizationName | string | This is the organization name to whom the customer belongs |
| 31 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 32 | customerRecordId | string | This is the bill to customer record id |
| 33 | mobileCustomerNumber | string | This is used by the mobile device to identify a customer. This is the objectId of customer. |
| 34 | mobileBillToNumber | string | This is used by the mobile device to the bill to customer. This is the objectId of customer. |
| 35 | mobileShipToNumber | string | This is used by the mobile device to identify the ship to customer. This is the objectId of customer. |
| 36 | Store Name | string | This is the name of the store where the items come from |
| 37 | Store Number | string | This is the store number where the items come from. |
| 38 | Notes | string | This is the notes section of the invoice |
| 39 | TransactionState | string | This can be paid, unpaid or refund |
| 40 | TransactionStatus | string | This can be paid, unpaid, refund, unknown |
| 41 | Payment Method | string | check, cash or credit card |
| 42 | Transaction Id | string | This is the id supplied by processor if payment method = credit card |
| 43 | Check Number | string | This is the check number if payment method = check |
| 44 | Reference Number | string | This is the same as check number and is used by quick books |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. | |
| 2 | Flag | string | “D” =this is an invoice detail item |
| 3 | objectId | string | This is the objectId for the detail if you are updating a record otherwise it is blank. |
| 4 | objectType | string | This is the objectType for the detail if you are updating a record otherwise it is blank |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp | |
| 6 | FunctionalGroupName | string | This is the group responsible for the record | |
| 7 | invoiceNumber | string | This number is set by the quickbooks service and is a unique positive number created by quickbooks |
| 8 | mobileInvoiceNumber | string | This number is created by the mobile device and is formed by combining the mobile unique device id-internal auto incrementing invoice |
| 9 | deviceId | string | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 10 | Quantity | Fractional # | This is the quantity ordered |
| 11 | itemNumber | string | This is the quickbooks item list number |
| 12 | Description | string | This describes the item |
| 13 | UM | string | Unit of measure normally blank |
| 14 | priceEach | Fractional # | This is the unit cost of the item |
| 15 | Amount | Fractional # | Quantity \* priceEach |
| 16 | Tax | String | This is the amount of tax on the item |
| 17 | shipToName | string | This is the person who will receive items |
| 18 | Customer:Job | string | This is the customer that will receive items and is selected from the customer list on mobile device |
| 19 | customerRecordId | string | This is the bill to customer record id |
| 20 | quantityPicked | Fractional # | This is the number of items picked from a store to fill the order and is filled out by shipping department. |
| 21 | quantityNotPicked | Fractional # | This is the number of items not picked from a store to fill the order and is filled out by shipping department. |
| 22 | quantityShipped | Fractional # | This is the actual quantity shipped filled out by shipping department |
| 23 | QuantityNotShipped | Fractional # | This is the number of items not shipped to customer and is filled out by shipping department |
| 24 | itemType | string | This is invoicedetail |
| 25 | organizationName | string | This is the organization name to whom the customer belongs |
| 26 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 27 | Store Name | string | This is the name of the store where the items come from |
| 28 | Store Number | string | This is the store number where the items come from. |

Each data line is separated by a CRLF. Data fields within a line are separated by a comma. All data elements are surrounded by quotes. You can have a header record and no details if you are doing an update on the header. To update a detail you must have the header record.

If the Invoice Numbrer or Mobile Invoice Number exist then you are doing an update operation otherwise you are doing a record creation.

Example of Curl command:

curl -k -X POST -F [media=@fields.txt](mailto:media=@fields.txt) [{webserver}/](https://www.rcofox.com/Image2000/rest/)quickbookservice/setInvoices/login/password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setSalesOrders

This function allows you to create/edit one or more sales orders. This can also be used for sales to associate an opportunity with any user. Note that we have a few fields like quantity not in quickbooks.

**POST:**

{webserver}/quickbookservice/setSalesOrders/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | Organization Name | string | This is the organization name to whom the customer belongs |
| 8 | Organization Number | string | This is the organization number to whom the customer belongs. |
| 9 | Sales Order Number | string | This number is set by the quickbooks service and is a unique positive number created by quickbooks |
| 10 | MobileSalesOrderNumber | string | This number is created by the mobile device and is formed by combining the mobile unique device id-internal auto incrementing invoice |
| 11 | Device Id | string | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 12 | Date | Date | This is the creation date of the invoice. |
| 13 | Purchase Order | string | This is the customer supplied purchase order |
| 14 | UserRecordId | string | This is when you do opportunity selling you can have a user and multiple oportunities |
| 15 | Ship To Company | string | This is the person who will receive invoice items |
| 16 | Ship To Name | string | This is the person who will receive invoice items |
| 17 | Ship To Address | string | This is the address of where items will get delivered. |
| 18 | Ship To CityStateZip | string | This is the city, state and zip information of where the items will get delivered by shipping carrier |
| 19 | shipToUserRecordId | string | This is the person who will receive invoice items |
| 20 | Bill To Company | string | This is the name of the person paying the invoice bill |
| 21 | Bill To Name | string | This is the name of the person paying the invoice bill |
| 22 | Bill To Address | string | This is the address of where items will get paid. |
| 23 | BillToCityStateZip | string | This is the city, state and zip information of where the items will get paid from. |
| 24 | billToUserRecordId | string | This is the address of where items will get paid. |
| 25 | Total | Fractional # | This is the invoice total amount |
| 26 | Message | string | This is the notes section of the sales order |
| 27 | Memo | string | This is the notes section of the sales order |
| 28 | Tax | string | Tax stuff |
| 29 | Tax Code | string | Tax stuff |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. | |
| 2 | Flag | string | “D” =this is an invoice detail item |
| 3 | objectId | string | This is the objectId for the detail if you are updating a record otherwise it is blank. |
| 4 | objectType | string | This is the objectType for the detail if you are updating a record otherwise it is blank |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp | |
| 6 | FunctionalGroupName | string | This is the group responsible for the record | |
| 7 | Organization Name | string | This is the organization name to whom the customer belongs |
| 8 | Organization Number | string | This is the organization number to whom the customer belongs. |
| 9 | Invoice Number | string | If the sales order gets converted to an invoice number then we can track the relationship |
| 10 | Mobile Invoice Number | string | This number is created by the mobile device and is formed by combining the mobile unique device id-internal auto incrementing invoice |
| 11 | Device Id | string | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 12 | Quantity | Fractional # | This is the quantity ordered |
| 13 | Item Number | string | This is the quickbooks item list number |
| 14 | Description | string | This describes the item |
| 15 | Amount | Fractional # | Quantity \* priceEach |
| 16 | Ordered | Fractional # | ??? |
| 17 | Rate | String | This is the amount of tax on the item |
| 18 | Tax | String | This is the amount of tax on the item |
| 19 | Master Barcode | string | This is the parent record barcode |
| 20 | Sales Order Type | string | Either service or item |
| 21 | Price Code | string | Used for discounting items |

Each data line is separated by a CRLF. Data fields within a line are separated by a comma. All data elements are surrounded by quotes. You can have a header record and no details if you are doing an update on the header. To update a detail you must have the header record.

If the Invoice Numbrer or Mobile Invoice Number exist then you are doing an update operation otherwise you are doing a record creation.

Example of Curl command:

curl -k -X POST -F [media=@fields.txt](mailto:media=@fields.txt) [{webserver}/](https://www.rcofox.com/Image2000/rest/)quickbookservice/setInvoices/login/password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setSalesTaxCodes

This function creates/updates sales tax codes. The attached csv file has the following format. The web services will check if the account coding field called “Sales Tax Code” exists and update coding fields otherwise a new account is created and the coding fields are set. The directory name is formed by using “Sales Tax Code”.

**POST:**

{webserver}/quickbookservice/setSalesTaxCodes/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | string | “H”-Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | salesTaxCode | string |  |
| 8 | salesTaxCodeInactive | string |  |
| 9 | Description | string |  |
| 10 | taxable | string |  |
| 11 | itemType | string | This is the record type |
| 12 | organizationName | string |  |
| 13 | organizationNumber | string |  |

Example of Curl command:

curl -k -X POST -F [media=@fields.txt](mailto:media=@fields.txt) [{webserver}/](https://www.rcofox.com/Image2000/rest/)quickbookservice/setSalesTaxCodes/login/password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setSalesTaxRates

This function creates/updates sales tax rates. The attached csv file has the following format. The web services will check if the account coding field called “Sales Tax Id” exists and update coding fields otherwise a new account is created and the coding fields are set. The directory name is formed by using “City, State”.

**POST:**

{webserver}/quickbookservice/setSalesTaxRates/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | string | “H”-Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | salesTaxId | string |  |
| 8 | salesTaxRate | string |  |
| 9 | city | string |  |
| 10 | state | string |  |
| 11 | itemType | string | This is the record type |
| 12 | organizationName | string |  |
| 13 | organizationNumber | string |  |

Example of Curl command:

curl -k -X POST -F [media=@fields.txt](mailto:media=@fields.txt) [{webserver}/](https://www.rcofox.com/Image2000/rest/)quickbookservice/setSalesTaxRates /login/password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | HUnique system wide user authentication string |
| Password | User’s password for authentication |

### setServices

This function creates/updates services. The attached csv file has the following format. The web services will check if the account coding field called “Number” exists and update coding fields otherwise a new account is created and the coding fields are set. The directory name is formed by using Number Name.

**POST:**

{webserver}/quickbookservice/setServices/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | string | “H”-Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | Item Number | string |  |
| 8 | Active | string |  |
| 9 | Subitem of | string |  |
| 10 | Rate | string |  |
| 11 | Tax Code | string |  |
| 12 | Contract Labor | string |  |
| 13 | Description | string |  |
| 14 | Task Description | string |  |
| 15 | itemType | string | This is the record type |
| 16 | organizationName | string |  |
| 17 | organizationNumber | string |  |

Example of Curl command:

curl -k -X POST -F [media=@fields.txt](mailto:media=@fields.txt) [{webserver}/](https://www.rcofox.com/Image2000/rest/)quickbookservice/setServices/login/password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setTerms

This function creates/updates invoice sales terms. The attached csv file has the following format. The web services will check if the account coding field called “Name” exists and update coding fields otherwise a new account is created and the coding fields are set. The directory name is formed by using Name which must be unique among names for this organization.

**POST:**

{webserver}/quickbookservice/setTerms/{login}/{password}/

**CSV File:**

**each Line terminated with CRLF**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Argument** | | **Data Type** | **Description** | |
| 1 | Operation | | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. | |
| 2 | HeaderDetailFlag | | string | “H”-Indicates this is a invoice header item | |
| 3 | objectId | | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. | | |
| 4 | objectType | | string | If invoice header exists then this is Invoice Header | | |
| 5 | MobileRecordId | String | | | This is used by the local device database and is generated by deviceid+timestamp | |
| 6 | FunctionalGroupName | string | | | This is the group responsible for the record | |
| 7 | Name | | string |  | |
| 8 | Term Type | | string | Standard or Date Driven | |
| 9 | Net Due In Days | | string | When is payment due typical net 5, net 10, net 30 | |
| 10 | Net Due Before the Nth Day of the Month | | string |  | |
| 11 | Discount Percentage | | string |  | |
| 12 | Discount If Paid Within Days | | string |  | |
| 13 | Due the Next Month If Issued Within N Days of Due Date | | string |  | |
| 14 | IsActive | | string | True/False indicate whether the term is active | |
| 15 | organizationName | | string |  | |
| 16 | organizationNumber | | string |  | |

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setTerms |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

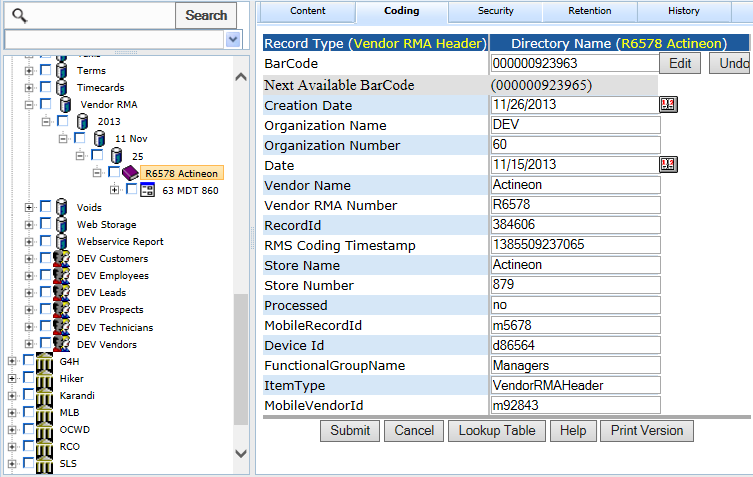
### setVendorRMAs

This function creates/updates vendor rma header and detail. The attached csv file uses a CRLF character to separate records. The csv file must contain at least one header line and zero or more detail lines. Each header line must start with an H character and each detail line must start with the D character and must contain the Invoice Number to link the detail and header record.

The current rule is that if a CSV row has non-blank ObjectId and ObjectType values, it the corresponding record will be updated.  If those values are blank, a new record will be created in the database.

There is a special case to point out:  if the mobile device wants to add a new Detail record under an existing Header record, in the CSV file you must include a row for the existing Header record with ObjectId and ObjectType, even if you do not need to update it -- then you can follow with any number of rows for new detail records.

The directory is formed by adding nodes under the Vendor RMA branch followed by year, month, day branches. Then you have your vendor rma header and vendor rma detail. The vendor rma header consists of the vendor rma number followed by a space character and then the vendor name. For the vendor rma detail the directory node name is composed of the item number followed by a space character and then the description.



**POST:**

{webserver}/quickbookservice/setVendorRMAs/{login}/{password}/

Example of Curl command:

curl -k -X POST -F [media=@fields.txt](mailto:media=@fields.txt) [{webserver}/](https://www.rcofox.com/Image2000/rest/)quickbookservice/setShipping/login/password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Method Name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | Device Id | string | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 8 | Date | Date | This is the creation date of the customer RMA. |
| 9 | Processed | string | The quickbooks service sets this rms coding value to processed after a synchronization |
| 10 | Vendor RMA Number | String | This is the rma supplied by vendor |
| 11 | ItemType | string | This is CustomerRMAHeader |
| 12 | Organization Name | string | This is the organization name to whom the customer belongs |
| 13 | Organization Number | string | This is the organization number to whom the customer belongs. |
| 14 | MobileVendorId | string | This is used by the mobile device to identify a vendorr. This is the objectId of vendor. |
| 15 | Store Name | string | This is the name of the store where the items come from |
| 16 | Store Number | string | This is the store number where the items come from. |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” =this is an invoice detail item |
| 3 | objectId | string | This is the objectId for the detail if you are updating a record otherwise it is blank. |
| 4 | objectType | string | This is the objectType for the detail if you are updating a record otherwise it is blank |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | DeviceId | string | Mobile device unique device id obtained from the rms when mobile device first initialized |
| 8 | Quantity to Return | Fractional # | This is the quantity to Return |
| 9 | Quantity Returned | Fractional # | Total number of items shipped to date |
| 10 | Quantity Shipped | Fractional # | The quantity just received |
| 11 | itemNumber | string | This is the quickbooks item list number |
| 12 | Description | string | This describes the item |
| 13 | Quantity Received | Fractional # | This is the number of items received |
| 14 | Vendor RMA Number | String | This is the factory rma number given to the customer and is auto-generated by the system |
| 15 | ItemType | string | This should be CustomerRMADetail |
| 16 | organizationName | string | This is the organization name to whom the customer belongs |
| 17 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 18 | Store Name | string | This is the name of the store where the items come from |
| 19 | Store Number | string | This is the store number where the items come from. |
| 20 | Notes | string | If return why item is returned |
| 21 | Processed | string | This shows the detail is finished |
| 22 | Quantity Not Shipped | string | This is the amount we have remaining to send to the vendor. |

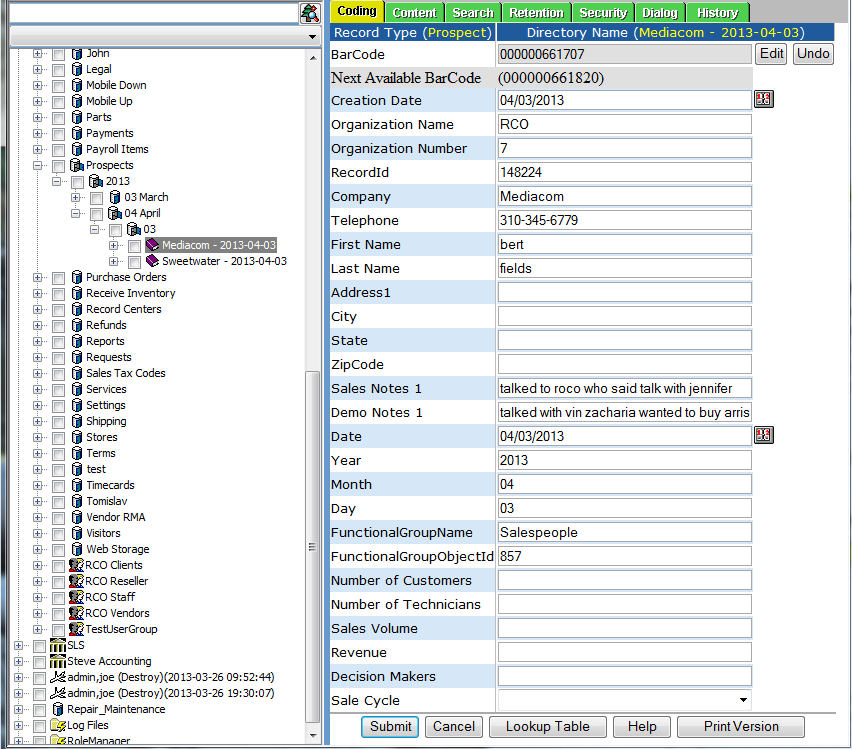
Each data line is separated by a CRLF. Data fields within a line are separated by a comma. All data elements are surrounded by quotes. You can have a header record and no details if you are doing an update on the header. To update a detail you must have the header record.

If the Invoice Numbrer or Mobile Invoice Number exist then you are doing an update operation otherwise you are doing a record creation.

## Sales Service

### setProspects

This function creates/updates prospects under the organization for the login under a storage container called Prospects. The attached csv file has the following format. The web services will check in the user login exists and update coding fields otherwise a new user is created and the coding fields are set. The directory name is formed by using Company and if Company is blank then Last Name, First Name.



**POST:**

{webserver}/salesservice/setProspects/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string |  |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | OrganizationName | string | Propects organization name |
| 8 | OrganizationNumber | string | Propects organization number |
| 9 | Date | Date | Date of the form and used for directory layout |
| 10 | LastName | string | Propects last name |
| 11 | FirstName | string | Propects first name |
| 12 | Title | string | Propects title ex. Dr. |
| 13 | Company | string | Propects company where they work |
| 14 | Address1 | string | Propects address1 where they work |
| 15 | Address2 | string | Propects address2 where they work |
| 16 | City | string | Propects city where they work |
| 17 | State | string | Propects state where they work |
| 18 | ZipCode | string | Propects postal zip code where they work |
| 19 | Country | string | Propects country where they work |
| 20 | Email | string | Propects email address |
| 21 | Telephone | string | Propects telephone number |
| 22 | Fax | string |  |
| 23 | Web Site | string |  |
| 24 | Sales Notes 1 | string |  |
| 25 | Demo Notes 1 | string |  |
| 26 | First Call Date |  |  |
| 27 | Number of Customers | string |  |
| 28 | Number of Technicians | string |  |
| 29 | Sales Volume | string |  |
| 30 | Revenue | string |  |
| 31 | Decision Makers | string |  |
| 32 | Sales Code | string | A=Hot B=medium |
| 33 | Salesperson | string |  |
| 34 | Job Start Date | string |  |
| 35 | Job End Date | string |  |
| 36 | Job Hardware Cost | string |  |
| 37 | Job Labor Cost | string |  |
| 38 | Job Net Gain | string |  |
| 39 | Job Gross Amount | string |  |
| 40 | Close Date | string |  |
| 41 | Lead Source | string |  |

Example of Curl command:

curl -k -X POST -F [media=@fields.txt](mailto:media=@fields.txt) [{webserver}/](https://www.rcofox.com/Image2000/rest/)userservice/setUsers/login/password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

## Security Services

The following security services are used to test whether logins are unique and can be used.

### getAssignedRoleNames

This module returns a list of role names for a given login, password.

**GET:**

{webserver}/securityservice/getAssignedRoleNames/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | isLoginUnique - action code or method name |
| loginId | login you want to test whether it is unique |
| Password | User’s password |

### getUserInfo

This module given login, password then the rms get the user’s first name, last name, object id, object type, user id.

**GET:**

{webserver}/securityservice/getUserInfo/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| loginId | login you want to test whether it is unique |
| Password | User’s password |

### getUserRights

This module gets all the rights for a given login, password.

**GET:**

{webserver}/securityservice/getUserRights/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| loginId | login you want to test whether it is unique |
| Password | User’s password |

### isAssignedRight

This module returns true or false and will tell you whether a given login, password has an assigned right.

**GET:**

{webserver}/securityservice/isAssignedRight/{login}/{password}/{right}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| loginId | login you want to test whether it is unique |
| Password | User’s password |
| Right | The name of the assigned right string you want to check |

### isAssignedRole

This module returns true or false and will tell you whether a given login, password has an assigned role.

**GET:**

{webserver}/securityservice/isAssignedRole/{login}/{password}/{role}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| loginId | login you want to test whether it is unique |
| Password | User’s password |

### isLoginUnique

This module tests if a logon string is already in use or is unique and can be used to create a new user.

**GET:**

{webserver}/securityservice/isLoginUnique/{loginId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| loginId | login you want to test whether it is unique |
| Password | User’s password |

### remoteSessionLogin

This module does a remote session and returns the sessionId.

**GET:**

webserver}/securityservice/remoteSessionLogin/{loginId}/{password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |

## Ship Services

### createBolHeader

This creates a bill of lading header record. A bill of lading is used as an inventory sheet of items to move from one location to another.

**POST:**

{webserver}/shipservice/createBolHeader/{login}/{password}/

**CSV File**

**Arguments**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| BillOfLadingNumber |  |
| ValidationDate |  |
| ValidationTime |  |
| ValidationInitials |  |
| ShippedFromShow |  |
| ShippedFromBoothNumber |  |
| CarrierName |  |
| Re-RouteMethod |  |
| Re-RouteAirTime |  |
| CarrierExhibitorInitial |  |
| SplitShipments |  |
| ShipToCompanyName |  |
| ShipToAddress |  |
| ShipToCity |  |
| ShipToState |  |
| ShipToCountry |  |
| ShipToZip |  |
| ShipToAttn |  |
| ShipToPhone |  |
| CarrierPayment |  |
| BillCompanyName |  |
| BillStreetAddress |  |
| BillCity |  |
| BillState |  |
| BillCountry |  |
| BillZip |  |
| BillAttn |  |
| BillPhone |  |
| RR# |  |
| ExhibitorOrAgent |  |
| Pro# |  |
| DateLoaded |  |
| LoadStartTime |  |
| LoadFinishTime |  |
| Carrier |  |
| Trailer# |  |
| PiecesReceived |  |
| DriverDate |  |
| DriverTime |  |

### createBillOfLadingDetail

This creates a detail Bill of Lading record under the Bill of Lading Header Record. There can be one or more detail records per header.

**POST:**

File=Name-Value.txt {webserver}/shipservice/createBillOfLadingDetail/{login}/{password}/

**CSV File**

parentObjectId, parentObjectType, checkerPieceCount, ExhibitorPieceCount, cartons, trunksCases,skidsPallets,carpets,carpetPadding,estimatedWeight

### createBillOfLadingSignature

This creates a bill of lading signature image record under the bill of lading header record.

**POST:**

File = signatureFile.png

{webserver}/shipservice/createBillOfLadingSignature/{login}/{password}/signatureDate}/{signatureTime}/{description}/{parentObjectId}/{parentObjectType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| parentObjectId | Authentication for login/password |
| signatureDate | Date of signature |
| signatureTime | Time of signature |
| Description | Decription of printed name |
| parentObjectType | Authentication for login/password |
| parentObjectType | Authentication for login/password |

### createShipHeader

This creates a ship header record. Under the organization number a node called Shipping must have been created.

**POST:**

{webserver}/shipservice/createShipHeader/{login}/{password}/{InvoiceNumber}/{VendorRmaNumber}/{carrier}/{trackNumber}/{destinationRecordId}/{mobileRecordId}/{customer:Job}/{MobileInvoiceNumber}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| InvoiceNumber | This is the invoice header recordId |
| VendorRma | this is the recordId of the vendor rma header form |
| Carrier | Name of the shipping company |
| trackNumber | This is the shipping companies tracking id |
| destinationRecordId | this is the customer store receiving record id |
| mobileRecordId | This is unique identifier created by mobile device |
| customer:Job | Customer job to match invoice |
| MobileInvoiceNumber | If you generate the shipping offline (not recommended) |

### createShipDetail

This creates a detail ship record under the ship Header Record. There can be one or more detail records per header.

**POST:**

{webserver}/shipservice/createShipDetail/{login}/{password}/{parentObjectId}/{parentObjectType}/{shippedItemRecordId}/{quantity}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| parentObjectId | This is the id of the item receipt parent |
| parentObjectType | This is the object type which should be Item Receipt Header |
| shippedItemRecordId | Part number can be non-tracked or tracked |
| quantity | Number of items needed by receiver |

### createShipSignature

This creates a ship signature image record under the ship header record.

**POST:**

File = signatureFile.png

{webserver}/shipservice/createShipSignature/{login}/{password}/signatureDate}/{signatureTime}/{description}/{parentObjectId}/{parentObjectType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| parentObjectId | Authentication for login/password |
| parentObjectType | Authentication for login/password |

### getBillOfLadingDetailIds

This function gets the bill of lading details ids of the bill of lading header record types for a specific form number.

**GET:**

{webserver}/shipservice/getBillOfLadingDetailIds/{login}/{password}/{parentObjectId/{parentObjectType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password |  |
| parentObjectId |  |
| parentObjectType |  |

### getBillOfLadingHeaderId

This is used to read the bill of lading hearder record id for a given bill of lading number.

**GET:**

{webserver}/shipservice/getShipHeaderId/{login}/{password}/{BillOfLadingNumber}

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| BillOfLadingNumber | This is a unique bill of lading form number that was assigned by the system |

### getBillOfLadingHeaderIds

This is used to read the bill of lading hearder record ids for a given date range. Note that if the start and end date are the same the caller wants all the form numbers for that day.

**GET:**

{webserver}/shipservice/getShipHeaderId/{login}/{password}/{startDate}/{endDate}

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| startDate | Start date where to search for bill of lading numbers |
| endDate | End date where to search for bill of lading numbers |

### getBillOfLadingSignatureIds

This gets all the signature ids for a given bill of lading form number.

**GET:**

{webserver}/shipservice/getShipSignatureIds/{login}/{password}/{parentObjectId}/{parentObjectType}

**ARGUMENTS:**

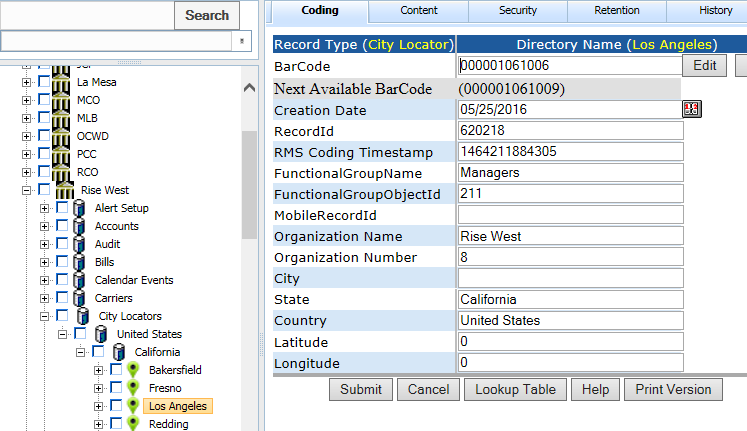
|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| parentObjectId | This is the parent object id of a task |
| parentObjectType | This is the parent record type |

### getCityLocator

This call will return a string that specifies the nearest city with distance and bearing. The bearing can be one of eight values (N, NE, E, SE, S, SW, W, NW). The distance is specified in miles.

Example: Fresno 10 miles NE

The directory structure is shown in the following figure. The directory name is the city which should be unique for each state.



**GET:**

{webserver}/shipservice/getCityLocator/{login}/{password}/{latitude}/{longitude}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| Latitude |  |
| Longitude |  |

### getHarvestCalendar

This call returns a matrix of date, variety, weight in pounds for the specified date range.

**GET:**

{webserver}/shipservice/getHarvestCalendar/{login}/{password}/{Start Date}/{End Date}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| Start Date |  |
| End Date |  |

### getEngineData

This call returns the odb2 engine data for the specified date. The date format is YYYYMMDD. The engine data files are stored in the following directory folder format.

EngineData

2015

08 August

15

20150815EngineData.csv

The csv file has the following form.

Timestamp,speed,odometer,brake,acceleration

**GET:**

{webserver}/shipservice/getEngineData/{login}/{password}/{date}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| Date | YYYYMMDD |

### getManifest

This call get the manifest at a given recordid. All records regardless of record type where ItemType=part and the LocationRecordId=RecordId are returned. The purpose of this call is for an operator to see all the customers that are getting orders (invoice number) and the details. Tracked parts show individually with s/n’s and non-track parts show with total quantity. If items are packed in boxes then the mobile can filter and only show the boxes and not the details for delivery people and it should remember the choice.

Examples

Costco Inv# 1234

Box 23456

Box 55678

Item# 45566 Qty: 34 Batteries

**GET:**

{webserver}/shipservice/getManifest/{login}/{RecordId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Logon | Unique string to identify user |
| password | User’s password |
| RecordId | This is the recordid of the object transporting customer orders |

### getNewShipHeaderIds

This is used to read new ship hearder record ids where the processed coding field = no where the record type is Ship Header

**GET:**

{webserver}/shipservice/getNewShipHeaderId/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |

### getShipDetailIds

This function gets the new invoice details ids of the ship header record types for a give parent ship header where processed = false.

**GET:**

{webserver}/shipservice/getShipDetailIds/{login}/{password}/{parentObjectId}/{parentObjectType}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| parentObjectId | This is the record id of the item receipt parent record |
| parentObjectType | This is the record type of the item receipt parent record which should be itemReceiptHeader |

### getShipHeaderId

This is used to read the ship hearder record ids for a given form number.

**GET:**

{webserver}/shipservice/getShipHeaderId/{login}/{password}/{formNumber}

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| formNumber | This is a unique shipping form number that was assigned by the RMS |

### getShipSignatureIds

This gets all the signature ids for a given ship form id.

**GET:**

{webserver}/shipservice/getShipSignatureIds/{login}/{password}/{parentObjectId}/{parentObjectType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| parentObjectId | This is the parent object id of a task |
| parentObjectType | This is the parent record type |
|  |  |

### getShipTrackStatus

This call will return the tracking numbers that have been received for the given carrier. We support the following cariers {Fedex, UPS, USPS, DHL}. The return arrary will be a csv file that has the following number

Tracking Number, Date, Time

**POST:**

{webserver}/shipservice/createBolHeader/{login}/{password}/{Carrier}

**CSV File**

**Arguments**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| TrackingNumber | Carrier Tracking Number |

### getShipWaybills

This call gets the information of the waybill header and waybill details given the recordid of the waybill header.

### getTimeAndDistance

This call will return the time in seconds and the distance in meters from your starting gps location (latitude, longitude) to your current gps location (latitude, longitude). This function is used to calculate whether you are overdue at a given location on a route. The calculations assume there is no traffic.

**GET:**

{webserver}/shipservice/getTimeAndDistance/{login}/{password}/{fromLatitude}/{fromLongitude}/{currentLatitude}/{currentLongitude}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| fromLatitude | This is the last from location latitude |
| fromLongitude | This is the last from location longitude |
| currentLatitude | This is your current location latitude |
| currentLongitude | This is your current location longitude |

### getTrackingInfo

This call returns the information from the tracking number supplied by the logistics carrier. Currently we support ups, usps, fedex and dhl.

**GET:**

{webserver}/shipservice/getTracking/{login}/{password}/{trackingnumber}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| trackingnumber | From the carrier |

### getTrailerData

This call returns the truck trailer sensor data for a given date. The date format is YYYYMMDD. There are 2 temperature sensors and 1 door sensor. The trailer data files are stored in the following directory folder format.

TrailerData

2015

08 August

15

20150815TrailerData.csv

The csv file has the following form.

Timestamp,temperature1,temperature2,doorStatus

**GET:**

{webserver}/shipservice/getTrailerData/{login}/{password}/{date}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| Date | YYYYMMDD |

### getTruckAlerts

This call returns all the alerts for a driver and can be filtered using optional start date and end date arguments.

**POST:**

{webserver}/shipservice/createTruckAlerts/{login}/{password}/{UserRecordId}/{Start Date}/{End Date}

**CSV File**

**Arguments**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| UserRecordId | Driver’s record id |
| Start Date | Optional start date for filtering |
| End Date | Optional end date for filtering |

### getTruckEngineDetails

This will return the concatenation of all the childrent content data files for a given trip and return as a single CSV file.

**GET:**

{webserver}/shipservice/getTruckEngineDetails/{login}/{password}/{TruckLogParentRecordId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| TruckLogParentRecordId | This is the parent truck log for a given trip normally current trip. |

### getTruckEngineNow

Get the most recent engine data read from the obd2 port. The data for the truck is stored in a folder on the truck server c:\TruckEngine. The truck engine data is stored every 10 seconds and currently we store the rpm. The file is engine\_2015-8-26.csv. The format of the data in the file is date,time,rpm

**GET:**

{webserver}/shipservice/getTruckEngineNow/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |

### getTruckGPSNow

Gets the most recent gps data. The data for the gps is stored in a folder on the truck server c:\gps. The gps data is stored every 10 seconds and currently we store the the following in 2 files LoggedData.csv and gpsnow.csv. The gpsnow.csv contains a header line and the last data record. The LoggedData.csv contains all the data from when the python program was started on the truck server.

LoggedData.csv & gpsnow.csv header format

"Count:","Date (dd/mm/yyyy):","Time (decimal seconds):","Latitude (decimal degrees):","Longitude (decimal degrees):","Velocity (miles per hour):","Current Bearing (degrees):","Total Distance (miles):", "Total Distance Today(miles):"

**GET:**

{webserver}/shipservice/getTruckGPSNow/{login}/{password}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |

### getTruckInfo

This will return the information for a given truck vehicle license number.

Latitude,Longitude,Temperature1,Temperature2,Door Status,Speed,Heading

**GET:**

{webserver}/shipservice/getTruckInfo/{login}/{password}/{VehicleLicensNumber}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| Vehicle License Number | Truck license number |

### getTruckLocationsFiltered

This will return the locations of all trucks in the following format. Note that we have a minimal return to keep data transfer minimal.

Vehicle license number,latitude,longitude

The filter is like a switch with n columns that can be turned on or off and the server will return the trucks that meet the filter criteria. Speeding is 80 miles per hour or greater and 39<Temp alert < 31. In the call you pass Booleans except from trip which is a string for the 3 list box choices. The speed and temp alert settings are defined for all trucks in the database settings and applies to all trucks. The temp2 and temp1 must be the same.

**GET:**

{webserver}/shipservice/getTruckLocationsFiltered/{login}/{password}/{doorclosed}/{temp2}/{temp1}/{speed}/{trip}/{overdue}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| Door Closed | True or False |
| Temp2 | True or False note that temp2 and temp1 have to the same |
| Temp1 | True or False |
| Speed | True or False |
| Trip | {trucksontrip,trucksnotontrip,alltrucks} |
| Overdue | Any truck at any location beyond overdue time limit |

### getTruckLogDetails

This will return the concatenation of all the children content data files for a given trip and return as a single CSV file.

**GET:**

{webserver}/shipservice/getTruckLogDetails/{login}/{password}/{TruckLogParentRecordId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| TruckLogParentRecordId | This is the parent truck log for a given trip normally current trip. |

### getTruckTemperatureDetails

This will return the concatenation of all the children content data files for a given trip and return as a single CSV file.

**GET:**

{webserver}/shipservice/getTruckTemperatureDetails/{login}/{password}/{TruckLogParentRecordId}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| TruckLogParentRecordId | This is the parent truck log for a given trip normally current trip. |

### getTruckSensorsNow

Gets the most recent gps summary data. The data for the gps is stored in a folder on the truck server c:\sensors. The sensor data is stored every 2 minutes and currently we store the the following information

MessageID,SensorID,Sensor Name,Date,Value,Formatted Value,Battery,Raw Data,Sensor State,GatewayID Alert Sent,Signal Strength,Voltage,Special Export Value

**GET:**

{webserver}/shipservice/getTruckSensorsNow/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |

### getTrucksInfo

This call will return an array of all the trucks info in the csv format show below.

recordId,objectid,objecttype,lat,lon,temp1,temp2,door status,heading,speed

**GET:**

{webserver}/shipservice/getTrucksInfo/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |

### getTrucksInGeofences

This call will return an array of all the trucks that are in danger and/or customer geofences.

GeofenceRecordId,Vehicle Serial Number, d or c, Latitud, Longitude

Where d = danger, c = customer}

**GET:**

{webserver}/shipservice/getTrucksGeofences/{login}/{password}/{danger}/{customer}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| danger | true=include danger geofences in search false if not |
| customer | true=include customer geofences in search false if not |

### getUpsTrackNumberInfo

This adds a UPS id to the track list. The Rms polls the ups service once an hour and when the package is delivered the rms will move all the parts in the store number = tracking id to the destination store. The receive date, receive time and receive by coding fields will get updated.

**GET:**

{webserver}/shipservice/getUpsTrackingNumberInfo/{login}/{password}/{trackingNumber}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| trackingNumber | This is the box tracking id supplied by the crrier. |

### setBillOfLadings

This call will create or update an existing bill of lading. The directory structure is Bill of Ladings, year, month, day (from shipdate). The directory name is BillToCompany

**GET:**

{webserver}/shipservice/setBilllOfLadings/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” =this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | organizationName | string | This is the organization name to whom the customer belongs |
| 8 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 9 | OrderNumber | string | This number is set by the quickbooks service and is a unique positive number created by quickbooks |
| 10 | CustomerNumber | string | This number is created by the mobile device and is formed by combining the mobile unique device id-internal auto incrementing invoice |
| 11 | Ship Date | Date | This is the date the invoice items were shipped and is filled out by shipping department using mobile def |
| 12 | Via | string | This is the name of the shipping carrier |
| 13 | Fob | string | This is to indicate freight on board for taxes |
| 14 | shipToCompany | string | This is the company who will pay invoice |
| 15 | shipToAddress | string | This is the address of where items will get delivered. |
| 16 | Ship To City | String |  |
|  |  |  |  |
| 17 | Ship To State | String |  |
| 18 | Ship To Country | String |  |
| 19 | Ship To Zipcode | String |  |
| 20 | Bill To Company | string | This is the name of the person paying the invoice bill |
| 21 | Bill To Address | string | This is the address of where items will get paid. |
| 22 | Bill To City | string | This is the city, state and zip information of where the items will get paid from. |
| 23 | Bill To State | string | This is the city, state and zip information of where the items will get paid from. |
| 24 | Bill To Country | string | This is the city, state and zip information of where the items will get paid from. |
| 25 | Bill To Zipcode | string | This is the city, state and zip information of where the items will get paid from. |
| 26 | Freight Charges Paid By | string | This number is supplied by the shipping carrier to track the invoice |
| 27 | Temperature Recorder Number | string | The quickbooks service sets this rms coding value to processed after a synchronization |
| 28 | USMC or DOT Number | string | This is a location id of where the item is going to |
| 29 | Carrier | string | This is invoiceheader |
| 30 | Trailer License Number | string | This is used by the mobile device to identify a customer. This is the objectId of customer. |
| 31 | Driver First Name | string | This is used by the mobile device to the bill to customer. This is the objectId of customer. |
| 32 | Driver Last Name | string | This is used by the mobile device to the bill to customer. This is the objectId of customer. |
| 33 | Received By | string | This is used by the mobile device to identify the ship to customer. This is the objectId of customer. |
| 34 | Purchase Order | string |  |
| 35 | ItemType | String |  |
| 36 | Seal Number | String |  |
| 37 | Receive DateTime | string |  |
| 38 | Truck Number | string |  |
| 39 | Notes | String |  |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | String | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | String | “D” - Indicates this is a detail item |
| 3 | objectId | String | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | String |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | String |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Order Number | String |  |
| 10 | Customer Number | String |  |
| 11 | Ship Date | Date |  |
| 12 | Quantity | Int |  |
| 13 | Item Number | String |  |
| 14 | Unit | String |  |
| 15 | Manufacturer Serial Number | String |  |
| 16 | Description | string |  |
| 17 | ItemType | string |  |
| 18 | Amount | String | This is the price |

### setBOLTruckRoute

This call will update the truck number in the Bill of Lading and Truck Route for a given order number.

**GET:**

{webserver}/shipservice/setBOLTruckRoute/{login}/{password}/{Order Number}/{Truck Number}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | sendSensorAlerts - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| Order Number | This is from accounting |
| Truck Number | Comes from the trailer loading |

### setShipWaybills

This call is used to create and edit waybills

**GET:**

{webserver}/shipervice/setShipWaybills/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

**Header record type is Ship Header see setShipParts!!!!!!**

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a detail item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Carrier | string | Dhl, fedex, ups, usps |
| 10 | TrackingNumber | string | This is the carier tracking number |
| 11 | Ship Date | string | YYYY-MM-DD this the date items were sent to destination |
| 12 | Ship Time | string | HH:MM:DD this is the time items were sent to destination |
| 13 | Receive Date | string | YYYY-MM-DD this the date items were received at destination |
| 14 | Receive Time | string | HH:MM:DD this is the time items were received to destination |

### getShipWaybillsNotProcessed

This call returns all the objectId, objectType, Carrier, Tracking Number of all waybills where processed = no.

### setCityLocators

This call will create one or more city locator records and assign the coding fields per the csv file.

**GET:**

{webserver}/shipservice/setCityLocators/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | City | string |  |
| 10 | State | string |  |
| 11 | Country | String |  |
| 12 | Latitude | String |  |
| 13 | Longitude | String |  |

### setMailHistory

This call creates/edits the mail history for a mail package. The directory layout is shown next. For the header you have the person’s name followed by their user record id. The detail names are composed of the date + time + sttaus

Mail History

Building Number Building Name

Year

Month

Day

Last Name, First Name 458686

2015-12-14 10:00:00 Out for Delivery

2015-12-14 10:30:00 Delivered

**GET:**

{webserver}/shipservice/setMailHistory/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Tracking Number | string |  |
| 10 | MailPackageRecordId | string |  |
| 11 | Status | String |  |
| 12 | Carrier | String |  |
| 13 | Building Name | String |  |
| 14 | Room Number | String |  |
| 15 | To Name | String |  |
| 16 | Date Received | Date |  |

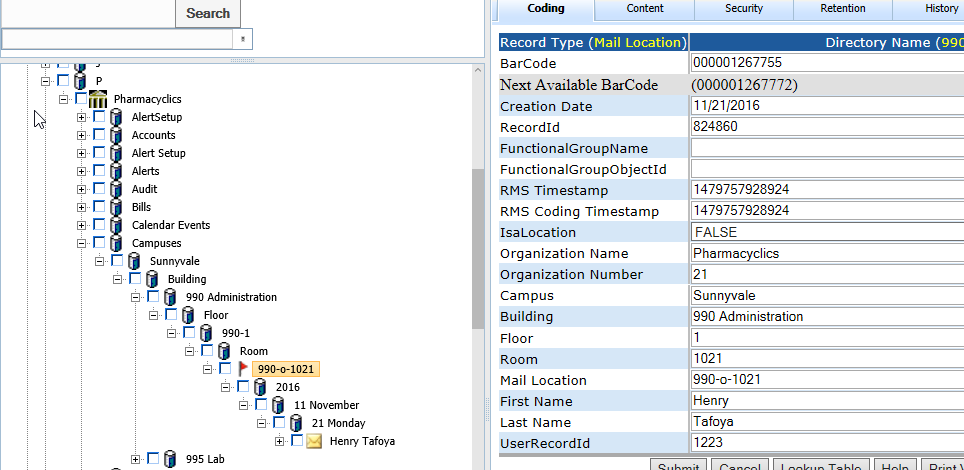
**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a detail item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Tracking Number | string |  |
| 10 | Status | string | Return to warehouse |
| 11 | Datetime | Date | 12/14/2015 |
| 12 | Location | String | Warehouse |
| 13 | LocationRecordId | String | 456694 |
| 14 | Mailcode | String | 991 |
| 15 | To Company | String |  |
| 16 | To Name | String |  |
| 17 | To Address | String |  |
| 18 | To State | String |  |
| 19 | To State | String |  |
| 20 | To Zipcode | String |  |
| 21 | To Country | String |  |
| 22 | Container Name | String | Mail Cart 1 |
| 23 | ContrainerRecordId | String | 43839 |
| 24 | MailPackageRecordId | string | 7788 |
| 25 | Carrier | string | 7788 |

### setMailLocations

This call creates mail delivery locations that are use in tracking the time and location of mail for an organization. The records are organized in the directory under a node called campuses then campus, building, floor then the room number (mail station). The directory record node name is the mail station. The operator pushes a cart with the mail and scans the mail location barcode label and all the mail going to that location.

Isalocation=true and set by the system during creation.



**GET:**

{webserver}/shipservice/setMailLocations/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | organizationName | string | This is the organization name |
| 8 | organizationNumber | string | This is the organization number |
| 9 | Campus | string |  |
| 10 | Building | string |  |
| 11 | Floor | String |  |
| 12 | Room | String |  |
| 13 | Mail Location | String |  |
| 14 | First Name | String |  |
| 15 | Last Name | String |  |
| 16 | UserRecordId | string |  |
| 17 | ItemType | string |  |

**Returns:**

Number of locations created/modified/deleted

Number+error message string

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setPickLists

A manager creates pick lists for one or more users by assigning them invoices to pick. Normally the manager will group invoices by location. The directory layout

Picklists

Year

Month

Day

PickerLastName, PickerFirstName PickerUserRecordid

Customer Invoice #

Customer Invoice #

**GET:**

{webserver}/shipservice/setPickLists/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | UserRecordid | string |  |
| 10 | FirstName | string |  |
| 11 | LastName | string |  |
| 12 | Date | Date |  |
| 13 | Time | string |  |
| 14 | Invoice Number | string |  |
| 15 | Customer Name | String |  |
| 16 | Customer Number | String |  |
| 17 | Customer ReccordId | String |  |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a detail item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | InvoiceNumber | String |  |
| 10 | Item Number | String | Also known as the SKU Stocking Unit |
| 11 | Description | string |  |
| 12 | FromLocationRecordId | String |  |
| 13 | ToLocationRecordId | String | When doing doing piece picking this is the tote the items are placed in |
| 14 | QuantityToPick | String |  |
| 15 | QuantityPicked | String |  |
| 16 | CustomerRecordId | String |  |
| 17 | FirstName | String |  |
| 18 | LastName | String |  |
| 19 | ManufacturerSerialNumber | String | This is a csv list of serial numbers. |
| 20 | Status | String | This is the pick status (picked, packed) |

### setShipLogs

This call use the record types ShipLogHeader and ShipLogDetail to create and or edit ship logs. Ship logs are used in the shipping industry to record the ports of call and AIS electronic information including the gps position, speed, course and area of operation.

**GET:**

{webserver}/shipservice/setShipLogs/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

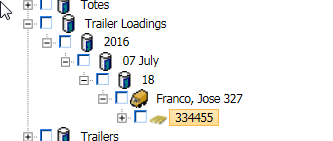
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Ship Name | string | As reported by AIS |
| 10 | IMO | string |  |
| 11 | MMSI | string | Phone number |
| 12 | Call Sign | string | Phone number |
| 13 | AIS Type | string | Driver Last Name |
| 14 | Gross Tonnage | string |  |
| 15 | Deadweight | string |  |
| 16 | Length | string |  |
| 17 | Breadth | string |  |
| 18 | Year Built | string |  |
| 19 | Active | Boolean | TRUE = On a trip |
| 20 | Info Received | Date |  |
| 21 | Area | String |  |
| 22 | Latitude | String |  |
| 23 | Longtitude | String |  |
| 24 | Status | String |  |
| 25 | Speed | String |  |
| 26 | Course | String |  |
| 27 | AIS Source | String |  |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a detail item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Destination | String |  |
| 10 | Last Known Port Name | String |  |
| 11 | Last Known Port Date | string |  |
| 12 | Previous Port Name | String |  |
| 13 | Previous Port Date | string |  |
| 14 | Max Speed Recorded | string |  |
| 15 | Average Speed Recorded | string |  |
| 16 | Draught | string |  |

### setTrailerLoadings

This call keep track of the trailer loadings for each truck. Records are stored under Trailer Loadings that are split by the Load Datetime. The header record is named using driver last name, driver first name then the truck number. The details are named using the pallet number.



**GET:**

{webserver}/shipservice/setTrailerLoadings/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

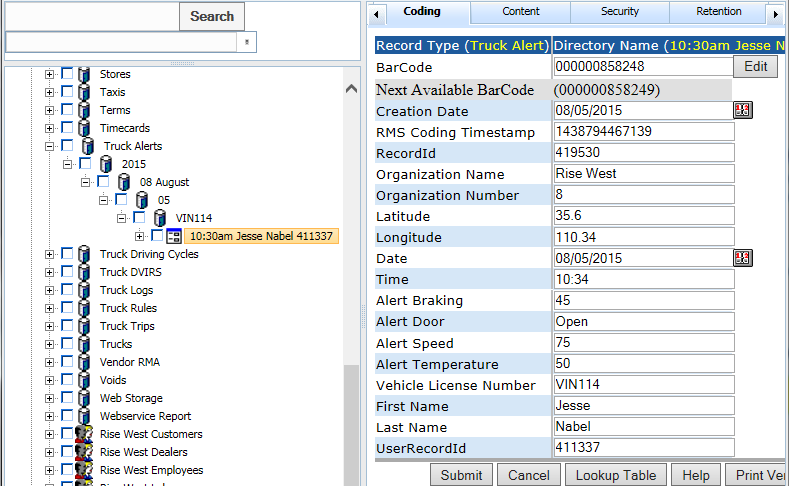
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Loader First Name | String |  |
| 10 | Loader Last Name | String |  |
| 11 | Loader RecordId | String |  |
| 12 | Driver First Name | String |  |
| 13 | Driver Last Name | String |  |
| 14 | DriverRecordId | String |  |
| 15 | Truck Number | String |  |
| 16 | Truck Route RecordId | String |  |
| 17 | Pallet Jack Number | String |  |
| 18 | Datetime | String | This the date and time the driver has to be at the location to start the delivery |
| 19 | Location | String | This is the place the driver needs to go and get the truck that will be used for the delivery |
| 20 | Trailer Number | String |  |
| 21 | LoadDateTime | string |  |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a detail item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Serial Numbers | String |  |
| 10 | Order Number | String |  |
| 11 | Invoice Detail RecordId | String |  |
| 12 | Customer Name | String |  |
| 13 | Customer Number | String |  |
| 14 | Location | String | This is the location in the truck |
| 15 | Items Total | String |  |
| 16 | Item Name | String |  |
| 17 | Item Number | String |  |
| 18 | Truck Route RecordId | String |  |
| 19 | LoadDateTime | String |  |
| 20 | Truck Number | String |  |
| 21 | Loader First Name | String |  |
| 22 | Loader Last Name | String |  |
| 23 | Loader RecordId | String |  |
| 24 | Driver First Name | String |  |
| 25 | Driver Last Name | String |  |
| 26 | Driver RecordId | string |  |
| 27 | Trailer1 Number | String |  |
| 28 | Trailer2 Number | string |  |

### setTruckAlerts

This call creates truck alerts based on one or more alert conditions. For example the driver was speeding and the back door flew open. The directory structure is shown below. Note that we do not record on the server the duration of the event and only the start of the event. All the long term data is stored on the mobile device (phone or laptop).



**GET:**

{webserver}/shipservice/setTruckAlerts/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | First Name | String |  |
| 10 | Last Name | String |  |
| 11 | UserRecordId | String |  |
| 12 | Date | String |  |
| 13 | Time | String |  |
| 14 | Latittude | String |  |
| 15 | Longitude | String |  |
| 16 | Alert Speed | String |  |
| 17 | Alert Brake | String |  |
| 18 | Alert Temperature | String |  |
| 19 | Alert Door | String |  |

### setTruckDVIRS

This call use the record types Truck DVIR Header and Truck DVIR Detail to create and or edit truck dvirs. Truck dvirs are used in the trucking industry to record the of truck inspections.

The DVIRS are store in the directory as follow

Truck DVIRS

Year

1st letter of last name

Last Name, First Name UserRecordId

Detail1

.

.

DetailN

**GET:**

{webserver}/shipservice/setTruckDVIRS/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Driver First Name | String |  |
| 10 | Driver Last Name | String |  |
| 11 | DriverUserRecordId | String |  |
| 12 | DateTime | String |  |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a detail item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Datetime | Date | Date of inspection |
| 82 | Location | string |  |
| 83 | Driver First Name | String | This is the driver |
| 84 | Driver Last Name | String |  |
| 85 | DriverRecordId | String |  |
| 86 | Vehicle License Number | String |  |
| 10 | Air Compressor | Boolean |  |
| 11 | Air Lines | Boolean |  |
| 12 | Battery | Boolean |  |
| 13 | Brake Accessories | Boolean |  |
| 14 | Brakes | Boolean |  |
| 15 | Carburetor | Boolean |  |
| 16 | Clutch | Boolean |  |
| 17 | Defroster | Boolean |  |
| 18 | Drive Line | Boolean |  |
| 19 | Fifth Wheel | Boolean |  |
| 20 | Front Axle | Boolean |  |
| 21 | Fuel Tanks | Boolean |  |
| 22 | Heater | Boolean |  |
| 23 | Horn | Boolean |  |
| 24 | Lights | Boolean |  |
| 25 | Mirrors | Boolean |  |
| 26 | Oil Pressure | Boolean |  |
| 27 | On Board Recorder | Boolean |  |
| 28 | Radiator | Boolean |  |
| 29 | Rear End | Boolean |  |
| 30 | Reflectors | Boolean |  |
| 31 | Safety Equipment | Boolean |  |
| 32 | Springs | Boolean |  |
| 33 | Starter | Boolean |  |
| 34 | Steering | Boolean |  |
| 35 | Tachograph | Boolean |  |
| 36 | Tires | Boolean |  |
| 37 | Transmission | Boolean |  |
| 38 | Wheels | Boolean |  |
| 39 | Windows | Boolean |  |
| 40 | Windshield Wipers | Boolean |  |
| 41 | Other | Boolean |  |
| 42 | Trailer1 Number | Boolean |  |
| 43 | Trailer1 Brake Connections | Boolean |  |
| 44 | Trailer1 Brakes | Boolean |  |
| 45 | Trailer1 Coupling (King) Pin | Boolean |  |
| 46 | Trailer1 Coupling Chains | Boolean |  |
| 47 | Trailer1 Doors | Boolean |  |
| 48 | Trailer1 Hitch | Boolean |  |
| 49 | Trailer1 Landing Gear | Boolean |  |
| 50 | Trailer1 Lights – All | Boolean |  |
| 51 | Trailer1 Other | Boolean |  |
| 52 | Trailer1 Roof | Boolean |  |
| 53 | Trailer1 Springs | Boolean |  |
| 54 | Trailer1 Tarpaulin | Boolean |  |
| 55 | Trailer1 Tires | Boolean |  |
| 56 | Trailer1 Wheels | Boolean |  |
| 57 | Trailer2 Number | Boolean |  |
| 58 | Trailer2 Brake Connections | Boolean |  |
| 59 | Trailer2 Brakes | Boolean |  |
| 60 | Trailer1 Coupling (King) Pin | Boolean |  |
| 61 | Trailer2 Coupling Chains | Boolean |  |
| 62 | Trailer2 Doors | Boolean |  |
| 63 | Trailer2 Hitch | Boolean |  |
| 64 | Trailer2 Landing Gear | Boolean |  |
| 65 | Trailer2 Lights – All | Boolean |  |
| 66 | Trailer2 Other | Boolean |  |
| 67 | Trailer2 Roof | Boolean |  |
| 68 | Trailer2 Springs | Boolean |  |
| 69 | Trailer2 Tarpaulin | Boolean |  |
| 70 | Trailer2 Tires | Boolean |  |
| 71 | Trailer2 Wheels | Boolean |  |
| 72 | Remarks | String |  |
| 73 | Condition Vehicle Satisfactory | Boolean |  |
| 74 | Drivers Signature Vehicle Satisfactory | string |  |
| 75 | Above Defects Corrected | Boolean |  |
| 76 | Above Defects No Correction Needed | Boolean |  |
| 77 | Mechanics Signature | string |  |
| 78 | Mechanics Signature Date | Date |  |
| 79 | Driver Signature No Corrections Needed | string |  |
| 80 | Driver Signature No Corrections Needed Date | date |  |
| 81 | Truck Number | string |  |
| 87 | Carrier | String |  |
| 88 | Address | String |  |
| 89 | Odometer | string |  |
| 90 | Mechanic First Name | String |  |
| 91 | Mechanic Last Name | String |  |
| 92 | MechanicRecordId | string |  |

**Signature Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D”=this is a invoice detail item |
| 3 | objectId | string | This is the timecard detail objectId. When you do an update you must have a header. |
| 4 | objectType | string | This is the timecard detail objectType |
| 5 | MobileRecordId | string | This is the group responsible for the record |
| 6 | FunctionalGroupName | String | This is used by the local device database and is generated by deviceid+timestamp |
| 7 | organizationName | string | This is the organization name to whom the customer belongs |
| 8 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 9 | Document Title | string |  |
| 10 | Issuing Authority | string |  |
| 11 | Expiration Date | Date |  |
| 12 | Signature Date | Date |  |
| 13 | Signature Name | String |  |
| 14 | Description | string |  |
| 15 | ItemType | string |  |
| 16 | Document Type | String |  |
| 17 | Document Date | Date |  |
| 18 | Reviewed By | String |  |
| 19 | Reviewed Date | Date |  |

### setTruckELDs

This call use the record types TruckELDHeader and TruckELDDetail to create and or edit truck logs. Truck logs are used in the trucking industry to record the hours of service and every driver must submit this at an inspection whether in writen or electronic format. Records are divided by year and month

The header is named by driver name cyclestartdate

The detail is named by record date

**GET:**

{webserver}/shipservice/setTruckELDs/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Vehicle License Number | String |  |
| 10 | CycleStartDateTime |  |  |
| 11 | Rule | String |  |
| 12 | Driver Name | String |  |
| 13 | DriverRecordId | string |  |
| 14 | DriverId | string |  |
| 15 | CoDriverName | string |  |
| 16 | CoDriverRecordId | string |  |
| 17 | CoDriverId | string |  |
| 18 | TruckLogHeaderRecordId | string |  |
| 19 | Truck Number | String |  |
| 20 | Trailer1 Number | String |  |
| 21 | Trailer2 Number | String |  |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a detail item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | 24-Hour Period Starting Time | String |  |
| 10 | Carrier Name | String |  |
| 11 | USDOT Number | string |  |
| 12 | Driver Name | string |  |
| 13 | Driver Id | String |  |
| 14 | DriverRecordId | string |  |
| 15 | Drivers License State | string |  |
| 16 | Co-Driver Name | string |  |
| 17 | Co-Driver Id | string |  |
| 18 | CoDriverRecordId | string |  |
| 19 | Current Location | string |  |
| 20 | Data Diagnostic Indicators | String |  |
| 21 | ELD Malfunction Indicators | string |  |
| 22 | ELD Registration Id | String |  |
| 23 | Unidentified Driver Records | String |  |
| 24 | Exempt Driver Status | String |  |
| 25 | Miles Today | String |  |
| 26 | Print-Display Status | String |  |
| 27 | Record Date | String |  |
| 28 | Shipping Id | String |  |
| 29 | Current Engine Hours | string |  |
| 30 | Engine Hours Start | string |  |
| 31 | Engine Hours End | string |  |
| 32 | Current Odometer | string |  |
| 33 | Odometer Start | string |  |
| 34 | Odometer End | string |  |
| 35 | Time Zone | string |  |
| 36 | Truck Number | string |  |
| 37 | Truck VIN | string |  |
| 38 | Trailer Number | string |  |
| 39 | TruckLogDetailRecordId | string |  |
| 40 | Off Duty Hours | string |  |
| 41 | Sleeper Hours | string |  |
| 42 | Driving Hours | string |  |
| 43 | On Duty Hours | string |  |
| 44 | Status | string |  |
| 45 | Comments | string |  |
| 46 | Rule | string |  |
| 47 | CycleStartDateTime | string |  |
| 48 | Trailer License Number | string |  |

### setTruckEngines

This call will create truckenginedetail record to an exisitng truck log header. The record type is TruckEngineDetail. The directory node name will be the date in the csv file. Please use the appendRecordContent to set the engine csv data to the node. You will use the I for the Operation to indicate that you want to create the directory node record.



**GET:**

{webserver}/shipservice/setTruckEngines/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Date | Number |  |

### setTruckGeofences

This call will create/edit a truck geofence area. The record type is “Geofence”. The type determines the type and color of the geofence. When you see danger this is not an area you want the truck going into. The purpose of the customer is to alert them that the product is on the way and close by. The geofence number is automatically assigned by the system. For now the directory node name is a storage container for the first letter of the geofence name and the node names are stored in alphabetic order.

Geofence

L

LAX

**GET:**

{webserver}/shipservice/setTruckGeofences/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Name | Number | This is the geofence name and must be unique |
| 10 | Latitude | Number | This is the latitude of the center of the geofence |
| 11 | Longitude | Number | This is the longitude of the center of the geofence |
| 12 | Radius | Number | This is the radius in miles of the geofence |
| 13 | Geofence Danger Email | String | This is a csv list of email addresses to send alert for the given geofence when truck enters this area |
| 14 | Geofence Danger SMS | String | This is a csv list of sms addresses to send alert for the given geofence when truck enters this area |
| 15 | Geofence Customer Email | String | This is a csv list of email addresses to send alert for the given geofence when truck enters this area |
| 16 | Geofence Customer SMS | String | This is a csv list of sms addresses to send alert for the given geofence when truck enters this area |

### setTruckLogs

This call use the record types TruckLogHeader and TruckLogDetail to create and or edit truck logs. Truck logs are used in the trucking industry to record the hours of service and every driver must submit this at an inspection whether in writeen or electronic format.

**GET:**

{webserver}/shipservice/setTruckLogs/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Off Duty Hours | Number | Number of off duty hours |
| 10 | Sleeper Hours | Number | Number of hours in the sleeper |
| 11 | Driving Hours | Number | Total number of hours driving |
| 12 | On Duty Hours | Number | Total number of on duty hours |
| 13 | HomeOfficeRecordId | string |  |
| 14 | Home Office Name | string |  |
| 15 | Home Office Phone | string | Phone number |
| 16 | First Name | string | Driver First Name |
| 17 | Last Name | string | Driver Last Name |
| 18 | UserRecordId | string |  |
| 19 | Year | string |  |
| 20 | Vehicle License Number | string |  |
| 21 | Driver | string |  |
| 22 | DriverRecordId | string |  |
| 23 | Co Driver | string |  |
| 24 | CoDriverRecordId | string |  |
| 25 | Rule | string | This is the drive cycle 8 days 80 hours |
| 26 | Rule Driving Days | string | this is the total number of days from the start of the driving cycle. |
| 27 | Active | Boolean | When a trip is first created this is set to true. When another new trip is created by the driver then the previous trip active coding field is set to false |
| 28 | Total Distance | Float | This is the current estimated distance from the start of the trip |
| 29 | Speed Violations | Integer | During each day you track the number of speed violations where the truck was exceeding 75 mph for more than 10 minutes. When you increment the detail you also increment the header |
| 30 | Geofence Violations | Integer | During each day you track the number of speed violations where the truck was exceeding 75 mph for more than 10 minutes. When you increment the detail you also increment the header |
| 31 | Start Date | Date | Start date of the trip |
| 32 | Start Time | String | Start time of the trip |
| 33 | End Date | String | End date of the trip |
| 34 | End Time | String | End time of the trip |
| 35 | ItemType | String |  |
| 36 | Trip Name | String | Describes the trip |
| 37 | Overdue Time Limit | String |  |
| 38 | RouteHeaderRecordId | String |  |
| 39 | HoursRemaining | String |  |
| 40 | Weight | String |  |
| 41 | Lot | String |  |
| 42 | Truck Number | String |  |
| 43 | Trailer1 Number | String |  |
| 44 | Trailer2 Number | String |  |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a detail item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Driving Hours | Fractional # | Hours of actual driving from gps truck movement |
| 10 | Shift Hours | Fractional # | Number of items needed by receiver |
| 11 | Time | string |  |
| 12 | Shift Reset | string |  |
| 13 | Cycle Hours | Fractional # | This is the serial number that was scanned if you are not using recordId’s |
| 14 | Cycle Type | string | Ex US 70hr/hday |
| 15 | Cycle Reset | string |  |
| 16 | Time Zone | string |  |
| 17 | Vehicle License Number | string |  |
| 18 | Truck Type | string |  |
| 19 | VIN | string |  |
| 20 | Load Description | string |  |
| 21 | Event Description | String |  |
| 22 | Event Date | string |  |
| 23 | Event Start | String |  |
| 24 | Event Duration | String |  |
| 25 | Event Status | String |  |
| 26 | Event Location | String |  |
| 27 | Event Notes | String |  |
| 28 | Carrier | String |  |
| 29 | Inspector | String |  |
| 30 | Inspection Notes | string |  |
| 31 | Start Date | string |  |
| 32 | Start Time | string |  |
| 33 | End Date | string |  |
| 34 | End Time | string |  |
| 35 | OffDuty | string |  |
| 36 | Sleeper | string |  |
| 37 | Driving | string |  |
| 38 | OnDuty | string |  |
| 39 | Driver | string |  |
| 40 | Co Driver | string |  |
| 41 | Equipment Info Numbers | string |  |
| 42 | Carrier Name | string |  |
| 43 | Carrier Address | string |  |
| 44 | Terminal | string |  |
| 45 | Rule | string |  |
| 46 | Total Miles this Cycle | string |  |
| 47 | Latitudes | string |  |
| 48 | Longitudes | string |  |
| 49 | Locations Descriptions | string |  |
| 50 | Total Miles Today | string |  |
| 51 | CoDriverRecordId | string | This will identify the co driver uniquely |
| 52 | DriverRecordId | string | This will identify the driver uniquely |
| 53 | Total Distance | Float | This is the current estimated distance from the start of the trip |
| 54 | Speed Violations | Integer | During each day you track the number of speed violations where the truck was exceeding 75 mph for more than 10 minutes. When you increment the detail you also increment the header |
| 55 | Geofence Violations | Integer | During each day you track the number of speed violations where the truck was exceeding 75 mph for more than 10 minutes. When you increment the detail you also increment the header |
| 56 | Active Driver | String | This is the person (driver or codriver) currently driving. |
| 57 | Shipment Info | String | What is a description of the load |
| 58 | ItemType | String |  |
| 59 | Overdue | String |  |
| 60 | From Latitude | String |  |
| 61 | From Longitude | String |  |
| 62 | To Latitude | String |  |
| 63 | To Longitude | String |  |
| 64 | Departure Time | String |  |
| 65 | Departure Date | String |  |
| 66 | Trip Number | String | Generated by mobile vehicle license number + date |
| 67 | RouteHeaderRecordId | String | This points to a route if selected |
| 68 | CycleStartDateTime | Date |  |
| 69 | CycleEndDateTime | Date |  |
| 70 | Truck Number | String |  |
| 71 | Odometer Start | String |  |
| 72 | Odometer End | String |  |
| 73 | Engine Hours | String |  |
| 74 | OriginType |  |  |

### setTruckRoutes

This call use the record types TruckRouteHeader and TruckRouteDetail to create a truck route with waypoints. Under the organization there is a directory node call Truck Routes which are separated by the year, month and day (from date) then the first letter of the driver last name coding field.

The directory node name of the header is Last Name, First Name UserRecordId

The detail node name is Customer

**GET:**

{webserver}/shipservice/setTruckRoutes/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

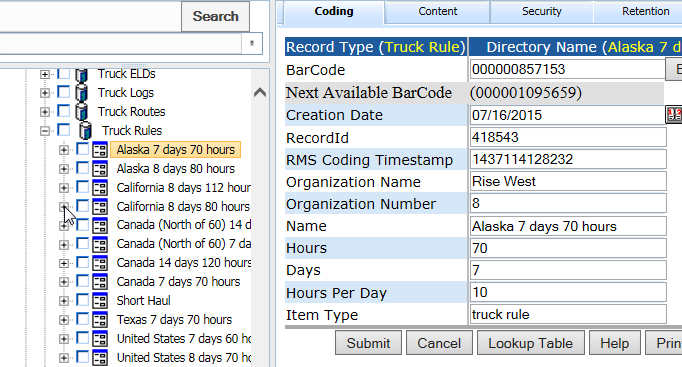
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Route Name | String | Name of the route or last name, first name userrecordid |
| 10 | Route Number | String | Number of the route |
| 11 | Driver First Name | String | Number of the route |
| 12 | Driver Last Name | String | Number of the route |
| 13 | DriverRecordId | String | Number of the route |
| 14 | ItemType | String | truckrouteheader |
| 15 | Date | String | This is scheduled date |
| 16 | Processed | string |  |
| 17 | Truck Number | string |  |
| 18 | Trailer1 Number | String |  |
| 19 | Trailer2 Number | String |  |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a detail item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String | Nu |
| 8 | Organization Number | String |  |
| 9 | Name | String | Name of the waypoint, customer name |
| 10 | Leg | String | Number of the route |
| 11 | Latitude | String |  |
| 12 | Longitude | String |  |
| 13 | Marker Type | String |  |
| 14 | ItemType | string | Truckroutedetailstop |
| 15 | Customer Name | string |  |
| 16 | Customer Number | String |  |
| 17 | CustomerRecordId | String |  |
| 18 | Order Number | String |  |
| 19 | BillOfLadingRecordId | String |  |
| 20 | Fist Name | String | This is the driver |
| 21 | Last Name | String |  |
| 22 | UserRecordId | string |  |
| 23 | Processed | string |  |
| 24 | Route Type | String | pickup or delivery |

### setTruckRules

This call create/edits truck rules and is stored under a node call Truck Rules using the name coding field for the directory node name.



**GET:**

{webserver}/shipservice/setTruckRules/{login}/{password}/

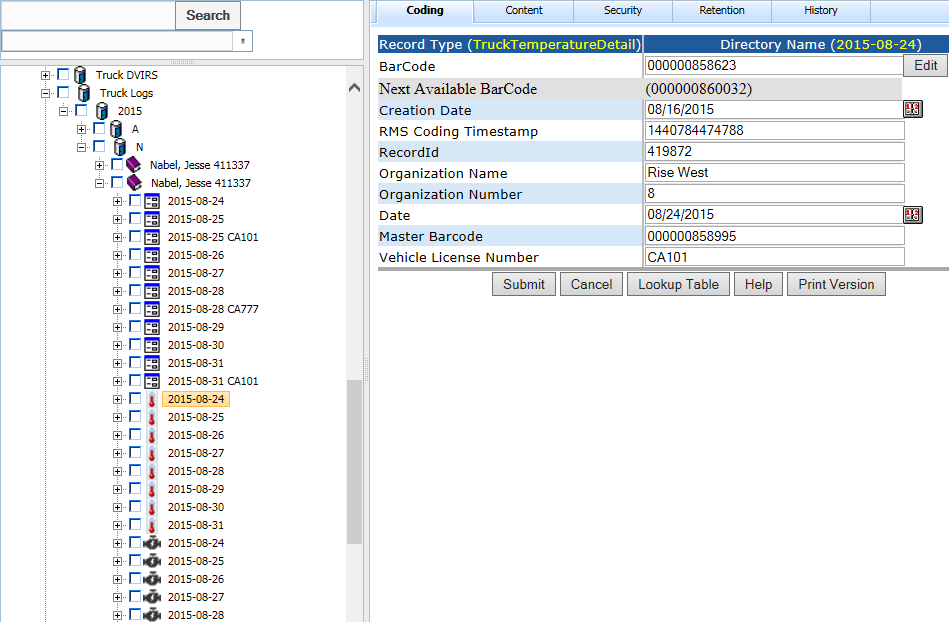
**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Name | String | Name of the route or last name, first name userrecordid |
| 10 | Hours | String | Number of the route |
| 11 | Days | String | truckrouteheader |
| 12 | Hours Per Day | String | This is the driver |
| 13 | ItemType | String | truckrule |

### setTruckSensors

This call will create trucktemperaturedetail record to an exisitng truck log header. Ther record type is TruckTemperatureDetail. There is one temperature log per day. The directory node name will be the date in the csv file. Please use the appendRecordContent to set the sensor csv data to the node. You will use the I for the Operation to indicate that you want to create the directory node record.



**GET:**

{webserver}/shipservice/setTruckSensors/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Date | Number |  |

### setTruckTrips

This call use the record types TruckTripHeader and TruckTripDetail to create and or edit truck trips. A truck trip provides detailed information on the truck waypoints for a given truck log.

**GET:**

{webserver}/shipservice/setTruckTrips/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

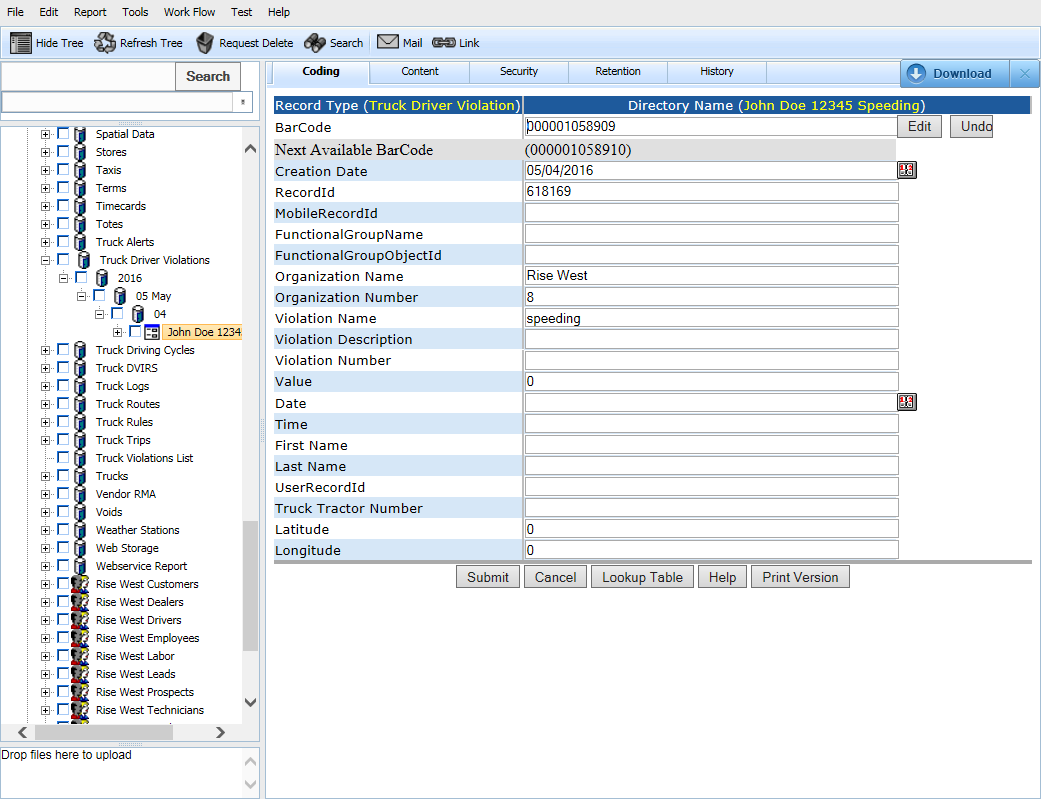
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Vehicle License Number | String |  |
| 10 | Trip Number | String |  |
| 11 | Driver | String |  |
| 12 | DriverRecordId | String |  |
| 13 | Active | string |  |
| 14 | Date | Date |  |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D” - Indicates this is a detail item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Trip Number | String | Unique number for this trip |
| 10 | From Location | string |  |
| 11 | From Latitude | Real# |  |
| 12 | From Longitude | Real# |  |
| 13 | To Location | String |  |
| 14 | To Latitude | Real# |  |
| 15 | To Longitude | Real# |  |
| 16 | Sales Order Number | string |  |
| 17 | Active | string |  |
| 18 | Leg | string |  |
| 19 | Departure Date | string |  |
| 20 | Departure Time | string |  |
| 21 | Distance | string |  |
| 22 | Via | String |  |
| 23 | Time without traffic | string |  |
| 24 | Time with traffic | String |  |

### setTruckDriverViolations

This call use the record type TruckViolation to create a new record. The record is stored under Truck Driver Violations then year, month and day. The directory record is named with First Name UserRecordId Violation Name.



**GET:**

{webserver}/shipservice/setTruckDriverViolations/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H” - Indicates this is a header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | ObjectType | string |  |
| 5 | MobileRecordId | String |  |
| 6 | FunctionalGroupName | string |  |
| 7 | Organization Name | String |  |
| 8 | Organization Number | String |  |
| 9 | Violation Number | String |  |
| 10 | Violation Name | String |  |
| 11 | Violation Description | String |  |
| 12 | Value | String |  |
| 13 | First Name | String |  |
| 14 | Last Name | String |  |
| 15 | UserRecordId | String |  |
| 16 | Truck Number | String |  |
| 17 | Latitude | String |  |
| 18 | Longitude | string |  |
| 19 | ItemType | String |  |
| 20 | DateTime | Date |  |

## SMS Service

### SmsMessageToFunctionalGroup

This sends an sms message to each member of the given functional group. The carrier is obtained by looking at the functional group member’s carrier coding field.

**GET:**

{webserver}/smservice/smsMessageToFunctionalGroup/{login}/{password}/{message}/{functional group name}/

**Returns:**

N – number of sms sent successfully

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| Message | Message to send |
| Functional group name | Name of the functional group you want to send message to |

## System Service

The following services deal with the health checks for the rms system on the server.

### setWebsiteMessage

This call logs the web site events and server will add a carriage return linefeed after each call.

The web services will update a file \\servername\cdrive\UploadWebsite\WebsiteEventLog.txt.

Date, Time, EventName, Description

Date, Time, EventName, Description

**GET:**

{webserver}/systemservice/setWebsiteMessage/{login}/{password}/{WebsiteMessage}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| logon | Unique string to identify user |
| password | User’s password |
| WebsiteMessage | This is a string describing a website message |

### getServerName

All connections that have an initial logon page will query the lion server for which server they use. For now the system will look at the login and if the first character is a \* then the server name returned = rcofox otherewise server name return = rcolion. If there is an error then -1 followed by an error string is returned.

**POST:**

{webserver}/systemservice/getServerName/{login}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| loginId | User’s login for authentication |

### getSystemStatus

This function determines whether the rms is operational and can access the file system and the database.

**POST:**

{webserver}/systemservice/getSystemStatus/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |

## Timecard Service

A user for a given organization works on one or more tasks. A task may have zero or more parts and photos. A timecard is created in the directory under an Organization under the forms node then under the Pending node and then under Year/Month/Day. Once the form is approved a call to get the approved timecard for a give date will then be sent to quick books where an invoice will be generated. If the user is a vendor (contractor) then an invoice will be generated so the vendor can get paid.

A user can only have one timecard per day!

### createTimecardHeader

This creates the timecard parent record with the coding data obtained from the user logon.

**POST:**

{webserver}/timecardservice/createTimecardHeader/{login}/{password}/{date}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| Date | Authentication for login/password |

### createTimecardTask

This creates the timecard parent record with the coding data obtained from the user logon.

**POST:**

{webserver}/timecardservice/createTimecardTask/{loginId}/{password}/{taskDate}/{taskTime}/{taskNumber}/{taskName}{parentObjectId}/{parentObjectType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| taskDate | This is the start date of the task |
| taskTime | This is the start time of the task |
| taskNumber | This is the service number from the accounting system |
| taskName | This is the service name from the accounting system |
| parentObjectId | This is the timecard parent id |
| parentObjectType | This is the timecard parent record type |

### createTimecardPart

This creates the timecard part record for a given task. The date

**POST:**

{webserver}/timecardservice/createTimecardPart/{login}/{password}/{itemNumber}/{description}{parentObjectId}/{parentObjectType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| parentObjectId | Authentication for login/password |
| parentObjectType | Authentication for login/password |

### createTimecardPhoto

This creates the timecard photo record.

**POST:**

{webserver}/timecardservice/createTimecardPhoto/{login}/{password}/photoDate}/{photoTime}/{description}/{parentObjectId}/{parentObjectType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| parentObjectId | Authentication for login/password |
| parentObjectType | Authentication for login/password |

### getTimecardsFiltered

This function returns all the timecards where the timecard header record coding field filtered by status. This function is for synchronizing the timecards on the mobile device.

|  |  |
| --- | --- |
| **State** | **Description** |
| In Progress | Locked by a mobile or web device for editing |
| Pending | This timecard has been submitted and is ready for review |
| Approved | The vendor timecard is ready for generating a vendor bill |
| Rejected | The timecard has been rejected |
| Open | The timecard is still being worked on |
| Closed | The vendor bill has been generated and no more editing allowed |
| Problem | The timecard has a problem and requires attention |

**GET**

{webserver}/timecardservice/getTimecardsFiltered/{login}/{password}/{status}/{maxtimestamp}

**Returns**

All Timecard Header Ids, Timecard Detail Ids and all the coding fields for each id

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| Status | This is the coding field you want to filter the timecards with. |
| MaxTimeStamp | The date of last time stamp stored on local system |

### getTimecardsApproved

This function returns all the timecards where the timecard header record coding field Status=Approved. This function is for paying the vendors that have done work. Note that the status has several states.

|  |  |
| --- | --- |
| **State** | **Description** |
| In Progress | Locked by a mobile or web device for editing |
| Pending | This timecard has been submitted and is ready for review |
| Approved | The vendor timecard is ready for generating a vendor bill |
| Rejected | The timecard has been rejected |
| Billed | The timecard has a problem and requires attention |
| Problem | The timecard has a problem and requires attention |

**GET**

{webserver}/timecardservice/getTimecardsApproved/{login}/{password}/

**Returns**

All Timecard Header Ids, Timecard Detail Ids and all the coding fields for each id

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |

### getTimecardsForInvoicing

This is used to read all the timecards that have been approved and either the customer billed status or the vendor billed status is “Not Billed”. This function is used to generate customer and vendor invoices. Note that vendor and customer invoicing can occur at different times and both have status fields called customer billed status and vendor billed status. There has to be at least one vendor in an organization. To submit a timecard you have to be a vendor or a staff member of the organization. Some timecards may not generate a bill if all the detail items are not billable.

The timecard status can have the following states

|  |  |
| --- | --- |
| **State** | **Description** |
| In Progress | The timecard is still being worked on by a mobile or web device |
| Pending | The timecard is awaiting local administrator to review |
| Approved | The timecard is ready for generating customer bills |
| Rejected | The timecard has been rejected |

The vendor billed status can have the following states

|  |  |
| --- | --- |
| **State** | **Description** |
| Billed | means the bill was created |
| Not Billed | means the bill was not created |
| Error | problem creating bill in accounting |

The customer billed status can have the following states

|  |  |
| --- | --- |
| **State** | **Description** |
| Billed | means the bill was created |
| Not Billed | means the bill was not created |
|  | No customer bill |
| Error | problem creating bill in accounting |

**GET**

{webserver}/timecardservice/getTimecardsForInvoicing/{login}/{password}/

**Returns**

All Timecard Header Ids, Timecard Detail Ids and all the coding fields for each id

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |

### getTimecardHeaderIds

This is used to read all the timecard ids for the status=approved.

**GET:**

{webserver}/timecardservice/getTimecardHeaderIds/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | getRecordMaxTimestamp - action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |

### getTimecardHeaderIdsByDateRange

This will get all timecard header ids in a date range for the login of the given organization.

**GET:**

{webserver}/timecardservice/getTimecardHeaderIdsByDateRange/{login}/{password}/{startdate}/{enddate}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| Startdate | Start of date range in YYYYMMDD format |
| enddate | end of date range in YYYYMMDD format |

### getTimecardHeaderIdsByStatus

This will get the timecard header ids using the status coding field (approved, pending, rejected, and billed) for the given organization.

**GET:**

{webserver}/timecardservice/getTimecardHeaderIdsByStatus/{login}/{password}/{status}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| Status | Text string for status coding field(approved, pending, rejected or processed) |

### getTimecardTotalHours

This function gets all timecard totals for a given user. The system will return 2 arrays (date, cumulative total hours). The time period default is the start of the year to the current date. This function is used in the timecard chart. When the Job Type for a user is not equal to “sales” then you do cumulative hours.

**GET:**

{webserver}/timecardservice/getTimecardTotalHours/{login}/{password}/{startDate}/{EndDate}

**ARGUMENTS:**

|  |  |
| --- | --- |
| Argument | Description |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| startDate | Default is start of current year |
| endDate | This is current date |

\* @param user – if calling the web service, this parameter is the login for the logged in user.

\* @param securityCode – if calling the web service, this parameter is the password for the logged in user.

\* @param strUserRecordIdOptional - if you know the timecard user's RecordId, pass it to save a lookup step.

\* @param userLoginOptional -- if you don't know the timecard user's RecordId but know the login,

\* pass it look up the user. If both user params are blank, the logged-in user will be the timecard user and used to

\* filter the data.

\* @param fromDate from-date filter value.

\* @param toDate to-date (inclusive) filter value.

\* @return returns a String[N][2] array, where x[N][0] contains the Date string, and

\* x[N][1] contains the Total string of the Nth timecard header record returned (0th being first).

### getTimecardSubmitters

This will get the pending timecard submitters for the login of the given organization and return the user recordIds, objectId, objectType of people submitting their timecards. Please not that you may have the same person with multiple timecards. Once a manager selects a submitter the program will get all pending timecards for that user.

**GET:**

{webserver}/timecardservice/getTimecardSubmitters/{login}/{password}/{status}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| status | Text string for status coding field(Approved, Pending, Rejected or In Progress) and if blank then Pending |

Returns: errorCode, treeId, objectId, objectType, name, parentTreeId, isEncrypted, encryptMode, scanCode, userLogin

### getTimecardTaskIds

This service gets all the tasks id for a given user and a given date.

**GET:**

{webserver}/timecardservice/getTimecardTaskIds/{login}/{password}/{date}/{parentObjectId}/{parentObjectType}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| parentObjectId | This is the id of the parent timecard that contains all the tasks |
| parentObjectType | This is the date of the approved user id’s timecard |
| Date | Date of the parent record |

### getTimecardPartIds

This gets all the part ids for a given task id for a given user and given date.

**GET:**

{webserver}/timecardservice/getTimecardPartIds/{login}/{password}/{parentObjectId}/{parentObjectType}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| objectId | This is the parent object id of a task |
| objectType | This is the parent record type |
| Date | This is the date of the approved user’s timecard |

### getTimecardNonBillableHours

This gets the non billable hours for given user for a given year.

**GET:**

{webserver}/timecardservice/getTimecardNonBillableHours/{login}/{password}/{year}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| Year | This is the year you want to get the total non-billable hours up to the current date |

### getTimecardYearToDateHours

This sets the total billable hours for given user for a given year from the beginning on the year up to the current date.

**GET:**

{webserver}/timecardservice/getTimecardYearToDateHours/{login}/{password}/{year}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| password | User’s password for authentication |
| Year | This is the year you want to get the total non-billable hours up to the current date |

### getUserTimecardsByStatus

This call gets all the user timecards for a given status string. The status string cannot be blank. Normally this call is used with status = in progress to get the timecard header and detail records that are getting saved to the rms from a user’s mobile device(s).

**GET**

{webserver}/timecardservice/getUserTimecards/{login}/{password}/{status}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| Status | The different status string are the following  In Progress – timecard currently being worked on  pending – a vendor timecard submitted for payment  approved – a timecard approved by local administrator  rejected – a timecard reject by local administrator |

### getUserTimecardsByStatusAndLogin

This call gets all the user timecards for a given status string and a given user login. The status string cannot be blank. Normally this call is used with status = in progress to get the timecard header and detail records that are getting saved to the rms from a user’s mobile device(s).

**GET**

{webserver}/timecardservice/getUserTimecardsByStatusAndLogin/{login}/{password}/{status}/{userlogin}/{fromDate}/{toDate}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| Status | The different status string are the following  In Progress – timecard currently being worked on  pending – a vendor timecard submitted for payment  approved – a timecard approved by local administrator  rejected – a timecard reject by local administrator |
| UserLogin | This is the login of the user you want to get data for. |
| fromDate | Start date of timecard in the system you want to find YYYYMMDD |
| toDate | End date of timecard in the system you want to find YYYYMMDD |

### setTimecardDetailAsCustomerBilled

The call will set the field Billed Status to Billed and will check if all details are now Billed. In case they are the call also changes the Header status to Billed. The accounting synchronization operations of customer and vendor bills are two separate commands.

**POST**

{webserver}/timecardservice/setTimecardDetailAsCustomerBilled/{login}/{password}/{objectId}/{objectType}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| objectId | This is the objectId for 1 detail record |
| objectType | This is the objectType for 1 detail record |

### setTimecardDetailAsVendorBilled

Each call will set the field Status to Billed and will check if all details are now Billed. In case they are the call also changes the Header status to Billed. The accounting synchronization operations of customer and vendor bills are two separate commands.

**POST:**

{webserver}/timecardservice/setTimecardDetailAsVendorBilled/{login}/{password}/{objectId}/{objectType}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| loginId | User’s login for authentication |
| Password | User’s password for authentication |
| objectId | This is the objectId for 1 detail record |
| objectType | This is the objectType for 1 detail record |

### setTimecards

This function creates/updates timecards. The attached csv file uses a CRLF character to separate records. The csv file must contain at least one header line and zero or more detail lines. Each header line must start with an H character and each detail line must start with the D character and must contain the Invoice Number to link the detail and header record.

**POST:**

{webserver}/timecardservice/setTimecards/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “H”= this is a invoice header item |
| 3 | objectId | string | If the timecard header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If timecard header exists then this is Timecard Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | String | This is the group responsible for the record |
| 7 | organizationName | string | This is the organization name to whom the customer belongs |
| 8 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 9 | Date | string | This is the date of the timecard |
| 10 | Email | string | This is the vendors email address |
| 11 | First Name | string | This is the vendor first Name |
| 12 | Last Name | string | This is the vendor last name |
| 13 | Company | string | This is the vendor’s company name |
| 14 | Memo | string | This is a memo field |
| 15 | Phone | string | This is the vendors telephone number |
| 16 | Reference Number | string | This is the vendor’s reference number |
| 17 | Status | string | This is a system field |
| 18 | Time | string | This is the time the timecard was submitted last |
| 19 | Total Hhours | string | This is the total number of hours of all detail records |
| 20 | Worker id | string | This is from the user coding RMS User Id |
| 21 | Due Date | Date | Date the timecard has to be paid by |
| 22 | Terms | String | This is the payment terms like net 30 |
| 23 | Customer Billed Status | String | This indicates wheter the customer bill in the accounting system has been created successfully  Billed – means the bill was created  Not Billed – mean no bill was generated  Error – problem creating bill in accounting |
| 24 | Vendor Billed Status | String | This indicates wheter the vendor bill in the accounting system has been created successfully  Billed – means the bill was created  Not Billed – mean no bill was generated  Error – problem creating bill in accounting |
| 25 | Employee Bill Status | String | This indicates wheter the employee bill in the accounting system has been created successfully  Billed – means the bill was created  Not Billed – mean no bill was generated  Error – problem creating bill in accounting |
| 26 | CreatorRecordId | String | Who filled out the timecard |
| 27 | Total Amount | Fractional Number | This indicates the sum of all the amounts in the timecard details |
| 28 | Processed | String |  |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | Flag | string | “D”=this is a invoice detail item |
| 3 | objectId | string | This is the timecard detail objectId. When you do an update you must have a header. |
| 4 | objectType | string | This is the timecard detail objectType |
| 5 | MobileRecordId | string | This is the group responsible for the record |
| 6 | FunctionalGroupName | String | This is used by the local device database and is generated by deviceid+timestamp |
| 7 | organizationName | string | This is the organization name to whom the customer belongs |
| 8 | organizationNumber | string | This is the organization number to whom the customer belongs. |
| 9 | Company | string | This is the vendor company |
| 10 | Customer:Job | string | This is vendor company or job name |
| 11 | First Name | string | Vendor first name |
| 12 | Last Name | string | Vendor last name |
| 13 | Hours | string | This is the number of hours for given task |
| 14 | Start Date | Date | This is the task start date |
| 15 | Start time | string | This is the task start time |
| 16 | End Date | Date | This is the task end date |
| 17 | End Time | string | This is the task end time |
| 18 | Task Detail | string | This describes what service or job done |
| 19 | Task Name | string | Same as service name |
| 20 | Task number | string | Service number |
| 21 | Worker Id | String | Vendor user RMS User Id |
| 22 | Billable | Boolean | Is the task a billable set of hours for vendor |
| 23 | CustomerRecordId | string | This is the customer the vendor did work for |
| 24 | Customer Billed Status | string | This indicates whether the customer bill (invoice) detail item has been generated.  Billed – means the bill was created  Not Billed – mean no bill was generated  Error – problem creating bill in accounting |
| 25 | itemType | string | This is TimecardDetail |
| 26 | Vendor Billed Status | string | This indicates whether the vendor bill detail item has been generated.  Billed – means the bill was created  Not Billed – mean no bill was generated  Error – problem creating bill in accounting |
| 27 | Employee Bill Status | string | This indicates whether the employee bill detail item has been generated.  Billed – means the bill was created  Not Billed – mean no bill was generated  Error – problem creating bill in accounting |
| 28 | CreatorRecordId | String | If a manager created a persons timecard |
| 29 | Overtime | Boolean | True – charge hours at overtime rate  False – charge hours at normal rate |
| 30 | Matter Number | String | This is the number identifying a matter |
| 31 | Matter Description | String | This is a description of the matter |
| 32 | Description | String | This is a description of a genral task |
| 33 | Amount | Fractional Number | This is the number of pieces a labor did in a single day. |
| 34 | Piece Rate | Fractional Number | Amount worker gets paid for each piece |
| 35 | Piece Quantity | Fractional Number | Total number of pieces worker produced/collected so far |
| 36 | Location | String |  |
| 37 | Labor Group Name | String |  |
| 38 | Labor Group Number | String |  |
| 39 | Overtime Hours | String |  |
| 40 | Double Overtime Hours | string |  |
| 41 | Straight Time Hours | String |  |
| 42 | Sick Time Hours | String |  |
| 43 | Holiday Time Hours | String |  |
| 44 | Vacation Time Hours | string |  |
| 45 | Weight | number |  |
| 46 | Employee Id | string |  |

Each data line is separated by a CRLF. Data fields within a line are separated by a comma. All data elements are surrounded by quotes. You can have a header record and no details if you are doing an update on the header. To update a detail you must have the header record.

If the Invoice Numbrer or Mobile Invoice Number exist then you are doing an update operation otherwise you are doing a record creation.

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setTimecardStatus

This sets the timccard status to Approved, Pending, Rejected, Billed.

**POST:**

{webserver}/timecardservice/setTimecardStatus/{login}/{password}/{objectId}/objecttype}/{status}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | action code or method name |
| login | User’s login for authentication |
| password | User’s password for authentication |
| objectId | This is the timecard object id of a task |
| objectType | This is the timecard record type |
| Status | This is a string {Approved, Pending, Rejected, Billed} |

## UserService

All the calls in this group deal with users. A user can only exist in one user group and their login is their email address in a hosted environment.

### authenticateUser

This function will return TRUE if user credential correct.

**POST:**

{webserver}/userservice/authenticateUser/{login}/{password}/

Returns: true of false

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| password | User’s password for authentication |

### createAlertUser

This creates a single alert user record under a staff id. There is a record type Alert Setup and the name of the directory node is Alert Setup must exist under the organization. If the staff user name does not exist the system will create the node. The alert users are stored under a storage group that starts with the first letter of their last name. Note that sometimes there are just companies and in this case the first letter of the company is used.

**POST:**

{webserver}/userservice/createAlertUser/{login}/{password}/{staffObjectId}/{staffObjectType}/{firstName}/{lastName}/{clientNumber}/{organizationMunber}/{/organizationName}/{email}/

Returns: true of false

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| password | User’s password for authentication |
| StaffObjectId | This is the person responsible for the customer |
| StaffObjectType | This is the object type (user) for the given staff member |
| firstName | First name of the alerted user |
| lastName | Last name of the alerted user |
| clientNumber | Client number of the alerted user |
| organizationNumber | Organization number associated with the alerted user |
| organizationName | Organization name of the alerted user |
| email | Email address of the alerted user |

### createUser

This restful interface is used to create a single user record.

**POST:**

{webserver}/userservice/createUser/{login}/{password}/{userGroupName}/{directoryName}/{firstName}/{lastName}/{loginClient}/{passwordClient}/{Role}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| userGroupName | User’s organizationName + “ “ + Clients, Staff, Vendors |
| directoryName | This is the rms directory name for the user |
| firstName | For an individual this is their first name |
| lastName | For an individual this is their last name, or company name |
| clientLogon | This is a unique client logon string |
| clientPassword | This is any password. |
| Role | User’s role default=NormalUser |

### createSecurityUser

This creates a security user under Security node. There is a record type Storage and the name of the directory node is Security must exist under the organization. If the corresponding customer user does not exist, it will create the customer user under ‘Organization Name + Customers’ and copy the coding field values over to the Security or Functional group node for the corresponding user. The Security users are stored under a storage group that starts with the first letter of their last name. Note that sometimes there are just companies and in this case the first letter of the company is used.

**POST:**

{webserver}/userservice/createSecurityUser/{login}/{password}/{userLogonId}/{userLogonPassword}/{firstName}/{lastName}/{clientName}/{clientNumber}/{company}/{address1}/{address2}/{city}/{state}/{zip}/{country}/{email}/{telephone}/{organizationNunber}/{/organizationName}

Returns: true of false

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| userLogonId |  |
| userPassword |  |
| firstName | First name of the alerted user |
| lastName | Last name of the alerted user |
| clientName | Client Name |
| client number |  |
| company |  |
| Address1 |  |
| Address2 |  |
| city |  |
| state |  |
| zip |  |
| country |  |
| email |  |
| telephone |  |
| organizationNumber | Organization number associated with the alerted user |
| organizationName | Organization name of the alerted user |

### createUserGroup

This call creates a single user group record. The SubOrganization Number and DivisionNumber are optional parameters. From the login you can get the Organization Number and if there are no SubOrganization Number and or Division Number then you create the user group under the organization number. If SubOrganization Number or Division Number does not exist you return an error and do not create a user group. The SubOrganization and Division are record types.

**POST:**

{webserver}/userservice/createUserGroup/{login}/{password}/{userGroupName}/{SubOrganizationNumber}/{DivisionNumber}/

RETURNS

ObjectId, ObjectType

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| userGroupName | User’s organizationName + “ “ + Clients, Staff, Vendors |
| SubOrganizationNumber | This is the suborganization number under the organization number |
| DivisionNumber | This is the division number under a suborganization number |

### deleteAlertUser

This removes an alerted user under an staff id. Deleting an alert user from RCOPublisher is disabled currently. However, this method is available and used by moveAlertUser call.

**POST:**

{webserver}/userservice/deleteAlertUser/{login}/{password}/{firstName}/{lastName}/

Returns: true or false

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| firstName | First name of the alerted user |
| lastName | Last name of the alerted user |

### deleteSecurityUser

This removes a security user under Security node. Deleting a security user from RCOPublisher is disabled currently. However, this method is available and used by moveSecurityUser call.

**POST:**

{webserver}/userservice/deleteSecurityUser/{login}/{password}/{firstName}/{lastName}

Returns: true or false

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| firstName | First name of the alerted user |
| lastName | Last name of the alerted user |

### deleteUser

This is used by a local administrator to delete a single user record. The caller has to supply all four arguments. The system will check that the login user is valid and that the userLogin is in the same organization number as the login.

**POST:**

{webserver}/userservice/createUser/{login}/{password}/{userLogin}/{userPassword}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| userLogin | This the login of the user you want to delete |
| userPassword | This the password of the user you want to delete |

### isUserLoginAvailable

This call returns true is the userLogin already exists false otherwise.

**GET:**

{webserver}/userservice/doesUserExist/{login}/{password}/{userLogin}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| userLogin | This is the user’s login (should be unique email address) |

### getCustomerAlertInfo

This returns customer alert information for a given first name and last name under a staff id.

**GET:**

{webserver}/userservice/getCustomerAlertInfo/{login}/{password}/{firstName}/{lastName}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| password | User’s password for authentication |
| firstName | First name of the alerted user |
| lastName | Last name of the alerted user |

Returns: NodeInfoBase object

### getFunctionalGroupMembers

This call returns all functional groups and their members (FirstName, LastName, UserRecordId, FunctionalGroupName and FunctionalGroupObjectId

**GET:**

{webserver}/userservice/getFunctionalGroupMembers/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| userLogin | This is the user’s login (should be unique email address) |

### getFunctionalGroupMaps

This call returns all functional group maps for a given login (organization number)

**GET:**

{webserver}/userservice/getFunctionalGroupMembers/{login}/{password}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| userLogin | This is the user’s login (should be unique email address) |

### getFunctionalGoupAlertInfo

This call returns user messaging information for all the users in a functional group.

**GET:**

{webserver}/userservice/getFunctionalGroupAlertInfo/{login}/{password}/{functionalGroupName}

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| FunctionalGroupName | Name of the functional group |

### getUserCoding

This call returns all the coding fields for a given userLogin.

**GET:**

{webserver}/userservice/getUserCoding/{login}/{password}/{userLogin}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| userLogin | This is the user’s login (should be unique email address) |

### getUserGroupMaps

This returns a list of node names or User Group names for employees and vendors configured in RCO system. If no user group name is defined for employees or vendors, the RCOPublish would use a default user group name as a node name for employees or vendors (ex. Organization name plus Employees -> SLS Employees for Employees or Organization name plus Vendors -> SLS Vendors for Vendors).

**GET:**

{webserver}/userservice/getUserGroupMaps/{login}/{password}

Returns: Map<String, String>

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |

### getUserId

This call is used to return the rms user id. The nodeInfoBase gives you lots of information but you can extract the objectId (userId) and objectType.

**GET:**

{webserver}/userservice/getUserId/{login}/{password}/{userLogin}/

RETURNS

nodeInfoBase

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| userLogin | This is the user’s login (should be unique email address) |

### getUsers

This call gets all the users and their associated coding fields from the Rms Coding Timestamp

**GET:**

{webserver}/userservice/getUsers/{login}/{password}/{maxTimeStamp}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| maxTimestamp | This is the maximum of content and content timestamps for all records of the given type. If not passed returns all records |

### getUserInfo

This call gets all the users and their associated coding fields from the Rms Coding Timestamp and login information.

**GET:**

{webserver}/userservice/getUserInfo/{login}/{password}/{maxTimeStamp}/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| maxTimestamp | This is the maximum of content and content timestamps for all records of the given type. If not passed returns all records |

### moveAlertUser

This moves an alert user from one node (first name/last name) to another node (new first name/new last name).

**POST:**

{webserver}/userservice/moveAlertUser/{login}/{password}/{staffObjectId}/{staffObjectType}/{firstName}/{lastName}/{newFirstName}/{newLastName}/{clientNumber}/{organizationMunber}/{/organizationName}/{email}/

Returns: true or false

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| StaffObjectId | This is the person responsible for the customer |
| StaffObjectType | This is the object type (user) for the given staff member |
| firstName | First name of the alerted user |
| lastName | Last name of the alerted user |
| newFirstName | New first name of the alerted user |
| newLastName | New last name of the alerted user |
| clientNumber | Client number of the alerted user |
| organizationNumber | Organization number associated with the alerted user |
| organizationName | Organization name of the alerted user |
| email | Email address of the alerted user |

### moveSecurityUser

This moves a security user from one node (first name/last name) to another node (new first name/new last name).

**POST:**

{webserver}/userservice/moveSecurityUser/{login}/{password}/{userLogonId}/{userLogonPassword}/{firstName}/{lastName}/{newFirstName}/{newLastName}/{clientName}/{clientNumber}/{company}/{address1}/{address2}/{city}/{state}/{zip}/{country}/{email}/{telephone}/{organizationNunber}/{/organizationName}

Returns: true of false

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| userLogonId |  |
| userPassword |  |
| firstName | First name of the alerted user |
| lastName | Last name of the alerted user |
| newFirstName | New first name of the alerted user |
| newLastName | New last name of the alerted user |
| clientName | Client Name |
| client number |  |
| company |  |
| Address1 |  |
| Address2 |  |
| city |  |
| state |  |
| zip |  |
| country |  |
| email |  |
| telephone |  |
| organizationNumber | Organization number associated with the alerted user |
| organizationName | Organization name of the alerted user |

### moveUser

This call is used to move a user from their current user group to a new user group.

**POST:**

{webserver}/userservice/moveUser/{login}/{password}/{objectIdUser}/{objectTypeUser}/{objectIdNewUserGroup}/{objectTypeUserGroup}/

RETURNS

Number+string (0+success or number+error message)

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| objectIdUser | This identifies the user |
| objectTypeUser | This is the user object type |
| objectIdNewUserGroup | This identifies the new user group |
| objectTypeNewUserGroup | This is the new user group object type |

### moveUserToNamedGroup

This call is used to move a user from their current user group to a new user group. Note that you must use the coding field called Record Name for the NamedGroup argument and you must match the case (case sensitive string). The purpose of this call is when you have a user in for example the Leads User Group and you want to move them to the Prospect User Group.

**POST:**

{webserver}/userservice/moveUserToNamedGroup/{login}/{password}/{UserObjectId}/{UserObjectType}/{NamedGroup}/

RETURNS

Number+string (0+success or number+error message)

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| UserObjectId | The record id of the user you want to move |
| UserTypeId | The record id of the user you want to move |
| NamedGroup | This is the new user group object type |

### renameUserGroup

This call is used to rename a single user record. An error will result if the new user group name exists. All user group names must be unique within an organization number.

**POST:**

{webserver}/userservice/renameUserGroup/{login}/{password}/{objectId}/{objectType}/{newUserGroupName}/

RETURNS

ObjectId, ObjectType

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| objectId | Id of the group you want to rename |
| objectType | Type of the group you want to rename |
| newUserGroupName | New user group name which must be unique |

### sendUserPassword

This call emails the user password to the login user looking and using their coding field call “Email”.

**POST:**

{webserver}/userservice/sendUserPassword/{login}/

RETURNS

True or False

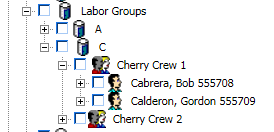
**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |

|  |  |
| --- | --- |
| Password | User’s password for authentication |

### setLaborGroups

This call will create labor groups and the members in the each labor group. The records are grouped under a storage container with the first letter of the group name and then the names are sorted by last name, first name UserRecordId.



**GET:**

{webserver}/userservice/setLaborGroups/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | organizationName | string | This is the organization name |
| 8 | organizationNumber | string | This is the organization number |
| 9 | Labor Group Name | string | Name of the labor group the person is a member of |
| 10 | Labor Group Number | string | Number of the labor group the person is a member of |
| 11 | Item Number | String | This is the current service number from services list |
| 12 | Description | string | This is a description of the current service |

**Detail Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | DetailFlag | string | “D” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | organizationName | string | This is the organization name |
| 8 | organizationNumber | string | This is the organization number |
| 7 | UserRecordId | string | Record id of member |
| 8 | First Name | First name of tjew | This is the user’s security role |
| 9 | Labor Group Name | string | Name of the labor group the person is a member of |
| 10 | Labor Group Number | string | Number of the labor group the person is a member of |
| 11 | Item Number | String | This is the current service number from services list |
| 12 | Description | string | This is a description of the current service |

**Returns:**

Number of Labor Groups Created

Number+error message string

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### setUserLogin

This call is used to set a user’s login.

**POST:**

{webserver}/userservice/setUserLogin/{login}/{password}/{objectIdUser}/{objectTypeUser}/{newLogin}/

RETURNS

ObjectId, ObjectType

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| objectId | Id of the user’s password you want to set |
| objectType | Object Type = user |
| newLogin | New user login (should be their email) |

### setUserPassword

This call is used to set a user’s password.

**POST:**

{webserver}/userservice/setUserPassword/{login}/{password}/{objectIdUser}/{objectTypeUser}/{newPassword}/

RETURNS

ObjectId, ObjectType

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |
| objectId | Id of the user’s password you want to set |
| objectType | Object Type = user |
| newPassword | New user password |

### setUserRole

This call is used to set a user’s role. Normally this is the string “NormalUser” and sometimes “LocalAdministrator”.

**POST:**

{webserver}/userservice/setUserRole/{login}/{password}/{objectIdUser}/{objectTypeUser}/{Role}/

RETURNS

0 – Success

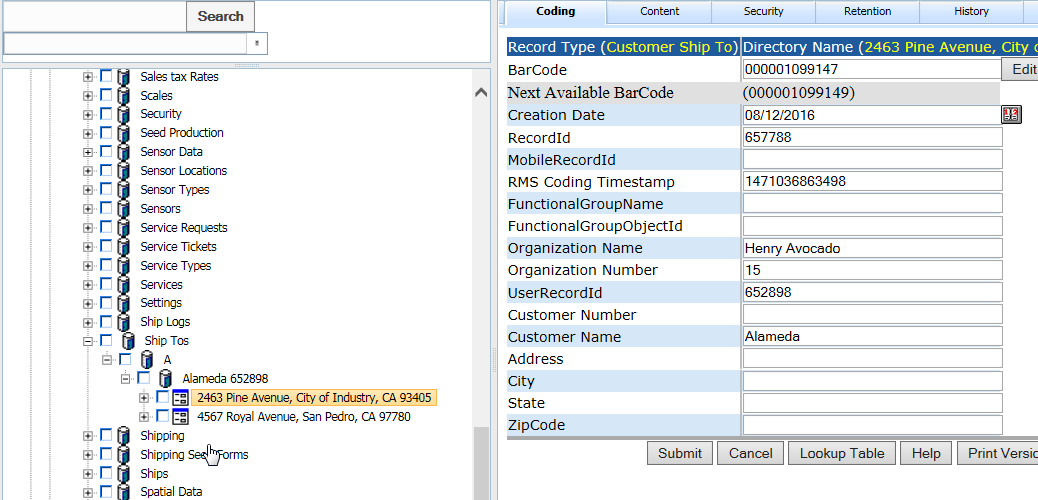
N – Error Message String

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| objectIdUser | Id of the user’s role you want to set |
| objectTypeUser | Object Type = user |
| Role | This is a string for the role and must exist in the rms |

### setUserShipTos

This function creates/updates user ship tos. The attached csv file has the following format. The directory name is under ShipTos and then using first letter of Company then company name and all the ship tos which are formed using the address, city, state zipcode.



**POST:**

{webserver}/userservice/setUserShipTos/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | DetailFlag | string | “D” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp |
| 6 | FunctionalGroupName | string | This is the group responsible for the record |
| 7 | organizationName | string | This is the organization name |
| 8 | organizationNumber | string | This is the organization number |
| 7 | UserRecordId | string | Record id of customer user |
| 8 | Customer Number | String |  |
| 9 | Customer Name | string |  |
| 10 | Address | string |  |
| 11 | City | string |  |
| 12 | State | String |  |
| 13 | Zipcode | string | This is a description of the current service |

### setUsers

This function creates/updates users. The attached csv file has the following format. The web services will check in the user login exists and update coding fields otherwise a new user is created and the coding fields are set. The directory name is formed by using first the Company and if this is blank then Last Name, First Name. If the user does not exist the role is set to Normal User.

**POST:**

{webserver}/userservice/setUsers/{login}/{password}/

**CSV File:**

**Header Line terminated with CRLF**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Argument** | **Data Type** | **Description** |
| 1 | Operation | string | "I" - insert the record without checking if it exists -- fastest.  "U" - update existing record.  A search by ObjectId/ObjectType, mobileRecordId, or secondary fields will be performed first.  Error if not found.  "O" - insert or update the record.  A search is always performed first by mobileRecordId or secondary fields.  Update if found, insert if not.  Slowest.  "D" - delete the record.  If used on a Header record, the details are automatically deleted also.  "N" - no operation.  May be useful if updating details without needing to update the header, so save unnecessary header update. |
| 2 | HeaderDetailFlag | string | “H” - Indicates this is a invoice header item |
| 3 | objectId | string | If the invoice header has already been created then this is the objectId for the header which means this is an update operation. If blank then create header. |
| 4 | objectType | string | If invoice header exists then this is Invoice Header |
| 5 | MobileRecordId | String | This is used by the local device database and is generated by deviceid+timestamp | |
| 6 | FunctionalGroupName | string | This is the group responsible for the record | |
| 7 | userGroupName | string | This is the name of the user group the login gets created and or modified |
| 8 | Role | string | This is the user’s security role |
| 9 | Login | string | This is the user’s login should be an email address |
| 10 | Password | string | This is the user’s login password |
| 11 | OrganizationName | string | User’s organization name |
| 12 | OrganizationNumber | string | User’s organization number |
| 13 | Company | string | User’s company where they work |
| 14 | LastName | string | User’s last name |
| 15 | FirstName | string | User’s first name |
| 16 | Address1 | string | User’s address1 where they work |
| 17 | Address2 | string | User’s address2 where they work |
| 18 | City | string | User’s city where they work |
| 19 | State | string | User’s state where they work |
| 20 | ZipCode | string | User’s postal zip code where they work |
| 21 | Country | string | User’s country where they work |
| 22 | Email | string | User’s email address |
| 23 | Telephone | string | User’s telephone number |
| 24 | ItemType | string | This can be client, staff or vendor |
| 25 | Latitude | string | This is the gps latitude |
| 26 | Longtitude | string | This is the gps longtitude |
| 27 | Driver License Number | String |  |
| 28 | Driver License State | string |  |
| 29 | Location | string |  |
| 30 | Customer Number | string |  |
| 31 | Date of Hire | String |  |
| 32 | Date of Birth | string |  |
| 33 | UserType | String |  |
| 34 | Employee Id | string |  |

Example of Curl command:

curl -k -X POST -F [media=@fields.txt](mailto:media=@fields.txt) [{webserver}/](https://www.rcofox.com/Image2000/rest/)userservice/setUsers/login/password/

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | setRecordCodingFields |
| Login | Unique system wide user authentication string |
| Password | User’s password for authentication |

### updateAlertUser

This updates coding field values for an alerted user. Note that if the first name or last name of an alerted user is changed, moveAlertUser should be used as first name and last name is also the node name of an alert user.

**POST:**

{webserver}/userservice/updateAlertUser/{login}/{password}/{staffObjectId}/{staffObjectType}/{firstName}/{lastName}/{newFirstName}/{newLastName}/{clientNumber}/{organizationMunber}/{/organizationName}/{email}/

Returns: true or false

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| firstName | First name of the alerted user |
| lastName | Last name of the alerted user |
| clientNumber | Client number of the alerted user |
| email | Email address of the alerted user |

### updateSecurityUser

This updates coding field values for a security user. Note that if the first name or last name

of a security user is changed, moveSecurityUser should be used as first name and last name is also the node name of a security user.

**POST:**

{webserver}/userservice/updateAlertUser/{login}/{password}/{userLogonId}/{userLogonPassword}/{firstName}/{lastName}/{newFirstName}/{newLastName}/{clientName}/{clientNumber}/{company}/{address1}/{address2}/{city}/{state}/{zip}/{country}/{email}/{telephone}/{organizationNunber}/{/organizationName}

Returns: true or false

**ARGUMENTS:**

|  |  |
| --- | --- |
| **Argument** | **Description** |
| Action | Action code or method name |
| login | Unique system wide user authentication string |
| password | User’s password for authentication |
| userLogonId |  |
| userPassword |  |
| firstName | First name of the alerted user |
| lastName | Last name of the alerted user |
| clientName | Client Name |
| client number |  |
| company |  |
| Address1 |  |
| Address2 |  |
| city |  |
| state |  |
| zip |  |
| country |  |
| email |  |
| telephone |  |
| organizationNumber | Organization number associated with the alerted user |
| organizationName | Organization name of the alerted user |