UH Development   
Webservices for CRM integration  
for O.H.G. – Phase 2 (Repairs) v1.0

February 10th 2012



# UH Development: CRM Webservices Development Phase 2

History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date Issued** | **Authors** | **Reason** | **Sections affected** |
| 1 | To 10/2/2012 | Stephen Shortall and Jeremy Riches | Phase 2 development work specification, broken out from the original main specification document and updated based on discussions to date of issue. | All |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Summary

This document details the development requirement for a set of new webservices to be added into Universal Housing, for the purpose of integrating with CRM systems.

The contents of this document are based upon a requirements document produced by Mark Beanland of One Housing Group dated 28th October 2011; subsequent analysis by Civica; and discussions that have taken place up to the date of issue of this version of the document, as detailed in the document history. The solution outlined in this document has been designed to fulfil the requirements for One Housing Group, but re-worked into a generic collection of webservices.

For the webservice approach, naming conventions, overviews etc, please refer to the Phase 1 specification document.

# Original Requirements Document from O.H.G.

(Copied and pasted)

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Input parameters (mandatory in bold)** | **Output parameters** | **Comments** |
| createOrder | **prop\_ref** string  house\_ref string  **job\_desc** string (=rq\_problem)  access\_details string(=rq\_access)  contact\_name string  phone\_number string (=rq\_phone)  caller\_name  caller\_desc  **reported\_date** (=repnotify\_dt)  **supplier** string (=sup\_ref)  **total\_cost** decimal  **priority** string (=task.priority)  issued DateTime  due\_date DateTime  completed DateTime  late\_comp\_reason string  **current\_status** string =(wo\_status)  invoice\_no string  confirmation\_order bool  cancel\_reason string  ***collection of tasks:***  **sequence\_no** int (=task\_no)  **job\_code** string  **units** (=est\_units)  trade string  **charge\_to** string (=rep\_type)  task\_text string  issued DateTime  completed DateTime  **current\_status** string (=task\_status) | wo\_ref string | For our purposes, we are not interested in requests, as we only ever have one order per request and hence have a 1:1 mapping. Therefore, although an entry would be created in rmreqst, this would not be obvious from the interface, which never references rq\_ref.  We do not have works orders that have different priorities for different tasks, so the priority should appear as if it is a property of the works order.  It should be possible to adjust the cost of the order independent of the tasks (the cost of which we do not require setting) using the total\_cost. This should be achieved by the use of MINVAL and REDUCE tasks. These tasks should be invisible, however, in that they are never returned as part of the collection of tasks and attempts to update them using the sequence number will fail.  The MINVAL / REDUCE task should have a sequence number of 0, -1 or a very large number to prevent them colliding with inputted sequence numbers.  Trade of the SOR item will be used unless trade is specified. Due date should be calculated from the priority unless it is specified. |
| updateOrder | **wo\_ref** string  prop\_ref string  house\_ref string  job\_desc string (=rq\_problem)  access\_details string(=rq\_access)  contact\_name string  phone\_number string (=rq\_phone)  caller\_name  caller\_desc  reported date (=repnotify\_dt)  supplier string (=sup\_ref)  total\_cost decimal  priority string (=task.priority)  issued DateTime  due\_date DateTime  completed DateTime  latecomp\_reason string  current\_status string =(wo\_status)  invoice\_no string  confirmation\_order bool  cancel\_reason string  ***optional collection of tasks:***  **sequence\_no** int (=task\_no)  job\_code string  units (=est\_units)  trade string  charge\_to string (=rep\_type)  task\_text string  issued DateTime  completed DateTime  current\_status string (=task\_status) |  | Fields need only be included if they are being updated.  If any tasks are included, these are dealt with differently depending on whether or not the sequence number already exists. If so, the task is being updated; if not, it is being added. For added tasks, mandatory fields are as per the createOrder service. |
| createInspection | **source\_wo\_ref** string  **insp\_reason** string  **inspector** string (=supplier)  **inspection\_type** string (=job\_code)  priority string  issued DateTime  completed DateTime  cancel\_reason bool  inspection\_text (=task\_text)  insp\_outcome string  **current\_status** string(=wo\_status)  ***optional collection detailing the inspection outcome for each task***  **sequence\_no** int  **posti\_result** string | **inspection\_ref** string (=wo\_ref) | Priority can default to the order priority.  The ‘inspection narrative’ and ‘post inspection result’ free text boxes in the Workflow processes do not seem to write to UHT. If they do, these would also be useful additions. |
| updateInspection | **inspection\_ref** string  insp\_reason string  inspector string (=supplier)  inspection\_type string (=job\_code)  priority string  issued DateTime  completed DateTime  cancel\_reason bool  inspection\_text (=task\_text)  insp\_outcome string  current\_status string(=wo\_status)  ***optional collection detailing the inspection outcome for each task***  **sequence\_no** int  **posti\_result** string |  |  |
| createVisit | **wo\_ref** string  **hadiary\_sid** int  **visit\_prop\_appointment** DateTime  **visit\_slot\_type** string  visit\_outcome string  visit\_comment string | **visit\_sid** int | It is assumed that it will be possible to get the details, e.g. property\_sid, visitor\_sid and visiting\_sid from the appropriate tables, if required.  Appropriate indexes must be set up for this, where they are missing.  It is felt that it would be best to use the diary, date and slot type as inputs (instead of visit\_sid) in case several people are trying to book a slot at the same time. |
| deleteVisit | **visit\_sid** int |  | This clears (i.e. zero) all values pertaining to the visit, rather than deleting the slot, thereby freeing it up be used again |
| createSlot | **hadiary\_sid** int  **visit\_prop\_appointment** DateTime  **visit\_slot\_type** string | **visit\_sid** int | This adds a new slot. |
| deleteSlot | **visit\_sid** int |  | This deletes the row from the database, providing it is not booked. |
| updateVisitOutcome | **visit\_sid** int  visit\_outcome string  visit\_comment string |  | Enters the visit outcome and comments if required. |
|  |  |  |  |

# Summary of Webservices

|  |  |  |  |
| --- | --- | --- | --- |
| **O.H.G. method** | **Code** | **Civica Method Name** | **Description** |
| createOrder | WR52 | CreateRepairWithOrder | Combines CreateRepairRequest, CreateRepairTask and CollateWorksOrder, to create a works order “all in one”. Note – this assumes only one order per request. |
| Additional lower level services wrapped into the above | | |
| WR51 | CreateRepairRequest | Create a new rmreqst record |
| WT51 | CreateRepairTask | Create a new rmtask record |
| WO51 | CollateWorksOrder | Have system create a new rmworder record based on repair tasks on a request |
| updateOrder | WO52 | UpdateWorksOrder |  |
| createInspection | WI55 | CreateInspectionCollateAndProcess |  |
| Additional lower level services wrapped into the above | | |
| WI51 | CreateInspectionTask |  |
| WI53 | CollateInspectionOrder | Have system create a new rmworder record based on inspection tasks on a request |
| WI54 | ProcessInspectionOutcome |  |
| updateInspection | WT52 | UpdateInspectionOrder |  |
| WI52 | UpdateInspectionTask |  |
|  |  |  |
| createVisit | WV51 | CreateAppointment |  |
| deleteVisit | WV53 | DeleteAppointment |  |
| updateVisitOutcome | WV62 | SetAppointmentOutcome |  |
| CreateDiarySlot | WS51 | CreateDiarySlot |  |
| DeleteDiarySlot | WS53 | DeleteDiarySlot |  |

# Detailed Webservice Requirements

## WR52: CreateRepairWithOrder

This service is to facilitate the creation of Works Order records, according to the requirements of O.H.G.

Background

In UHT, repair requests are raised which creates an rmreqst record on the “Diagnose Repair Form” (Reactive repairs menu). The user will select the property the repair is to be carried out at and add tasks (rmtask records) that require to be completed. Upon the print order button being selected, the system will “Collate” these tasks into a “Works Order” (rmworder record). Depending on system settings all the tasks may be collated onto a single order or they may be split onto multiple orders. There may be a “Minimum order” value configured which represents the minimum value that should be on an order. If the value of the tasks doesn’t add up to that amount a “dummy” task is added to the request \ order with a value to make the order up to the minimum allowable.

Webservice Requirements / Assumptions

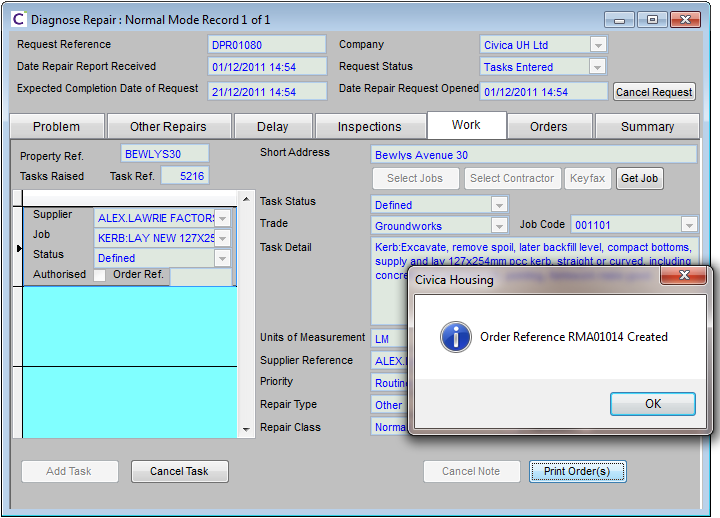
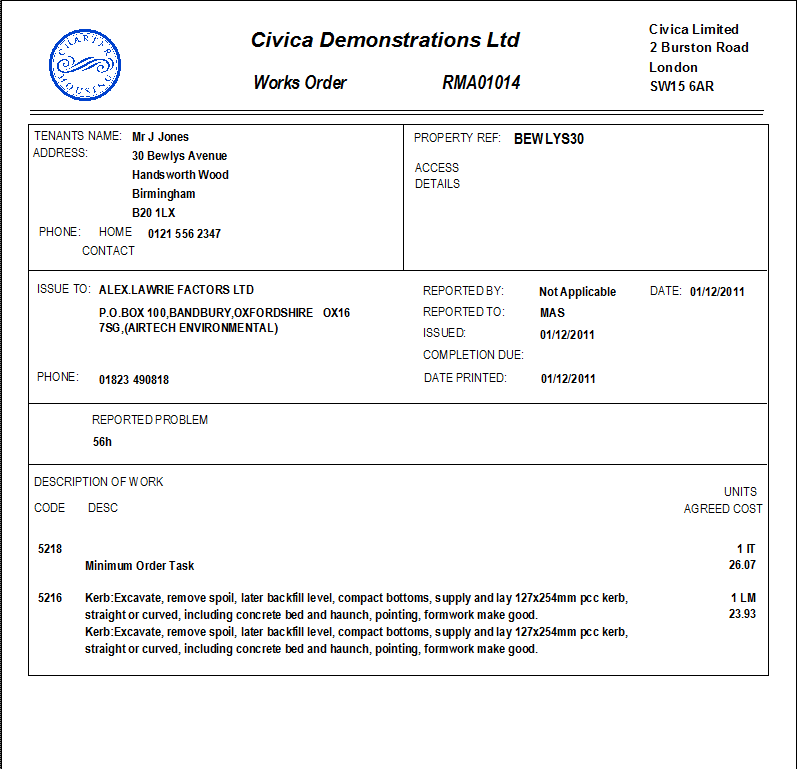
* 1:1 relationship between request and works order (i.e. a request only ever has one works order).
* The priority for each task on the works order is the same.
* Unless specified on the tasks passed in to the webservice, the trade should be derived from the SOR
* Unless specified by the data passed in to the webservice, the due date should be calculated from the priority.

O.H.G. have requested:  
  
*“It should be possible to adjust the cost of the order independent of the tasks (the cost of which we do not require setting) using the total\_cost. This should be achieved by the use of MINVAL and REDUCE tasks. These tasks should be invisible, however, in that they are never returned as part of the collection of tasks and attempts to update them using the sequence number will fail.*

*The MINVAL / REDUCE task should have a sequence number of 0, -1 or a very large number to prevent them colliding with inputted sequence numbers.”*

Having investigated the MINVAL tasks at OHG, they are created by the standard minimum order value functionality and so will automatically be covered by the generic solution.

We initially thought there may have been some bespoke work that had been done that was generating the REDUCE tasks during the collation process. Having checked it’s doesn’t look like there is any bespoke work here. Dave Cook in the Leeds office believes these REDUCE tasks are simply manually added by the users. If this is correct and they are currently doing it through the UH front end then they will be able to do the same thing with the CreateRepairTask method. If there is any bespoke business logic to be followed e.g. a REDUCE task should get generated if the task exceeds a certain value then I would suggest this is done client side. This way OHG will be able to maintain their own bespoke business rules if they change.

Separate methods will be created to replicate the UHT functionality of creating a repair request, adding a repair task to a repair request and collating the tasks on a request into an order. A method will be created for OHG that combines these three processes so that they can be achieved in a single web service call that takes a NewRepairDto as a parameter. This is in turn made up of a RepairRequestDto and a collection of RepairTaskDto  
  
  
  
  


Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
  + Data Object: NewRepairDto
    - (see Appendix A)
* Response
  + SOAP response for success/failure (SOAP-ENV fault tag)
  + Wo\_ref (works order reference)

Processing

Creation of repair request record in UHT table: rmreqst along with associated task records in table: rmtask. Creation of works order record on table: rmworder.

Errors

TBA

## WR51: CreateRepairRequest

This service is to facilitate the creation of repair request records following the existing UHT business logic

Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
  + Data Object: RepairRequestDto
    - (see Appendix A)
* Response
  + SOAP response for success/failure (SOAP-ENV fault tag)

Processing

Validation will be performed to ensure fields not available to the user in UHT are not set on the DTO. A new repair request record will be created as per the existing UHT business logic.

Errors

TBA

## WT51: CreateRepairTask

This service is to facilitate the creation of new a repair task

Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
  + String: RequestReference
  + Data Object: RepairTaskDto
    - (see Appendix A)
* Response
  + Sreinf: TaskReference
  + SOAP response for success/failure (SOAP-ENV fault tag)

Processing

A new rmtask record will be created based on the DTO and linked to the specified repair request record. The generated task\_ref of the new record will be returned.

Errors

## WO51: CollateWorksOrder

## Collates repair tasks on a request into a works order as per the existing UHT business logic

Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
  + String: RepairRequestReference
* Response
  + Reference or the works order created
  + SOAP response for success/failure (SOAP-ENV fault tag)

Processing

Repair tasks on the request specified that have not been collated into an order yet will be collated following the existing UHT business logic

Errors

TBA

## WO52: UpdateWorksOrder

This service is to facilitate the update of Works Order records, following the existing UHT business logic

Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
  + Data Object: WorksOrderDto
    - (see Appendix A)
* Response
  + SOAP response for success/failure (SOAP-ENV fault tag)

Processing

Validation will be performed to ensure read only fields are not altered. WorksOrderDto will have a key of wo\_ref \ WorksOrderReference. The matching record in rmworder will be updated as per the existing UHT business logic. If there is no such record an error will be returned.

Errors

TBA

## WI55: CreateInspectionCollateAndProcess

This service is to facilitate the creation of Works Inspection records, according to the requirements of O.H.G.

Background

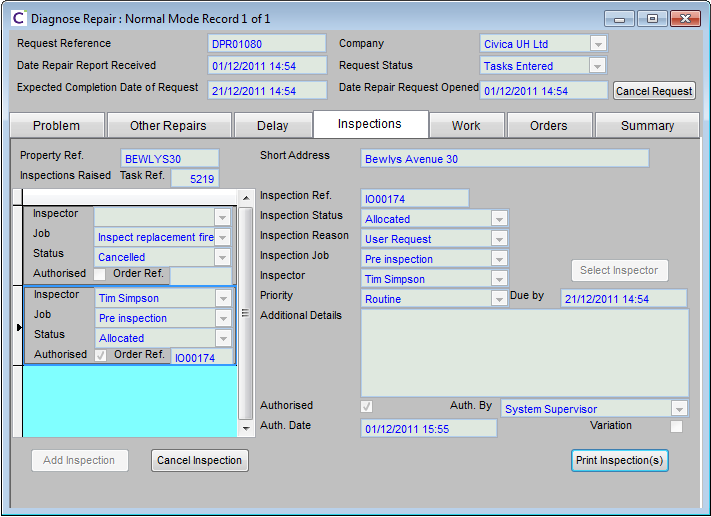
In UHT: Pre, Post and WIP Inspections can be raised. Inspections consist of inspection tasks and inspection orders just like normal repairs and are stored against the same tables – rmtask \ rmworder. In UHT the tasks are added first and when the print inspections button is pressed they get collated into an inspection order by the system.

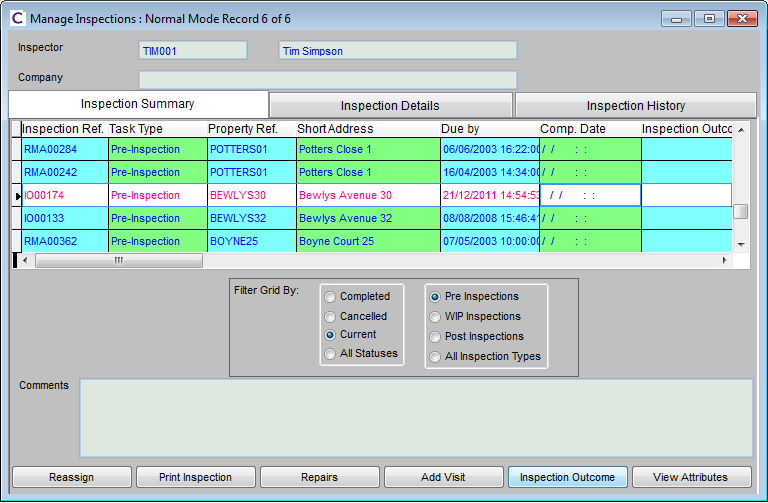
The separate functions will be written for the 3 processes of adding an inspection task, collating the order and completing the inspection which mirror the UHT functionality. This service will call the 3 functions one after another so the client can achieve all steps in a single service call.

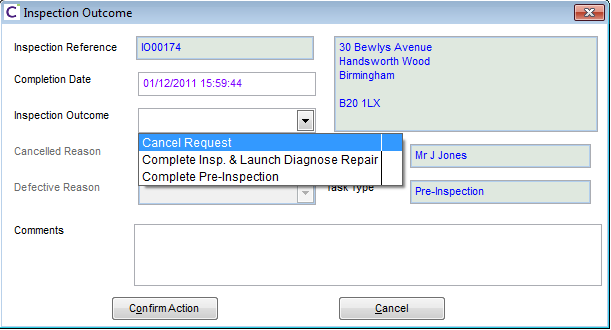
O.H.G. have requested that the “Inspection narrative” and “post Inspection result” free text boxes in the workflow processes would be useful additions if they write into UHT.

Webservice Requirements / Assumptions

* The Inspection order should be associated with a repair request, from which the priority should be derived. (Note: the OHG spec infers that the Inspection should be linked to a regular works order – in UHT they are linked to their parent repair request in the first instance. Once inspections are collated into an inspection order the tasks do get stamped with that orders wo\_ref).







Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
    - Data Object: RepairTaskDto, InspectionOutcomeDto(see Appendix A)
* Response
  + SOAP response for success/failure (SOAP-ENV fault tag)
  + Inspection\_Wo\_ref (inspection order reference)

Processing

Creation of inspection record in UHT table: rmworder along with associated task records in table: rmtask.

Errors

TBA

## WI51: CreateInspectionTask

This service is to facilitate the creation of a new inspection task

Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
  + Data Object: InspectionTaskDto
    - (see Appendix A)
* Response
  + String: Task Reference
  + SOAP response for success/failure (SOAP-ENV fault tag)

Processing

Similar to WT51 but an inspection task will be created rather than a regular repair task. The normal UHT business logic will be respected.

Errors

TBA

## WI53: CollateInspectionOrder

## This service is to facilitate the collation of Inspections tasks on the specified repair request into an inspection order

Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
  + String: RequestReference
* Response
  + String: WorksOrderReference
  + SOAP response for success/failure (SOAP-ENV fault tag)

Processing

Inspection tasks on the specified repair request that have yet to be collated onto an order will be collated as they would if the “Print Inspection(s)” button were clicked on the Inspections tab of the Diagnose repair form.

Errors

TBA

## WI54: ProcessInspectionOutcome

## This service is to facilitate the completion of an inspection order

Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
  + Data Object: InspectionOutcomeDto
    - (see Appendix A)
* Response
  + SOAP response for success/failure (SOAP-ENV fault tag)

Processing

The key of the InspectionOutcomeDto will be wo\_ref \ WorksOrderReference. The specified works order will be completed with the specified inspection outcome details. This will mirror the UHT functionality invoked by the “Confirm Action” from the “Inspection Outcome” form.

Errors

TBA

## WI52: UpdateInspectionTask

This service is to facilitate the update of Works Inspection records, and inspection outcomes according to the requirements of O.H.G.

Note: There can be logic of the back of an inspection completion. E.g. if the inspector decided no work needs to be done - the request may be cancelled.

Webservice Requirements / Assumptions

* See WI51

Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
  + Data Object: InspectionTaskDto
    - (see Appendix A)
* Response
  + SOAP response for success/failure (SOAP-ENV fault tag)

Processing

Update of inspection task record in UHT table: rmworder along with associated task records in table: rmtask.   
  
The key of the InspectionTaskDto will be task\_ref \ TaskReference. This rmtask record will be updated with the changes specified in the DTO. Validation will be performed to ensure read-only fields do not get updated. The existing UHT business logic will be followed. If there is no such record in rmtask a SOAP fault will be returned.

Errors

TBA

## WT52: UpdateInspectionOrder

## This service is to facilitate the updating of an inspection order

Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
  + String: RequestReference
  + Data Object: InspectionOrderDto
    - (see Appendix A)
* Response
  + SOAP response for success/failure (SOAP-ENV fault tag)

Processing

The key of the InspectionOrderDto will be wo\_ref \ WorksOrderReference. The related record in rmworder will be updated as per the existing business logic in UHT. If there are any changes to read-only fields a SOAP fault will be returned. If there is no related record in rmworder again a SOAP fault will be returned.

Errors

TBA

## WV51: CreateAppointment

This service is to facilitate the creation of Visit/appointment records, according to the requirements of O.H.G.

Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
  + Data Object: DiaryAppointmentDto
    - (see Appendix A)
* Response
  + SOAP response for success/failure (SOAP-ENV fault tag)
  + Visit\_Sid (unique reference)

Processing

This will create a diary appointment as per the existing UHW business logic. The visit record will be checked that it relates to an empty slot (visit\_slot = true and visit\_type = 3). If this check is satisfied the type will be set to 1 (booked) and the appointment fields will be set.

Errors

TBA

## WV53: DeleteAppointment

This service is to facilitate the removal of a vist/appointment, according to the requirements of O.H.G.  
  
The visit\_sid will be checked that it relates to an appointment rather than a visit or a slot before the record is converted back to its “free slot” state.

Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
  + Visit\_Sid
* Response
  + SOAP response for success/failure (SOAP-ENV fault tag)

Processing

The visit record will be checked so see if it related to an appointment (visit\_slot=true and visit\_type=1) This service should clear (zero-ise) all values pertaining to the visit, rather than deleting the slot, thereby freeing it to be used again. The visit\_type will be set back to 1 (free slot).

Fields un-set:

Errors

TBA

## WV62: SetAppointmentOutcome

This service is to facilitate the update of a visit outcome, according to the requirements of O.H.G.  
  
Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
  + Data Object: DiaryAppointmentOutcomeDto
    - (see Appendix A)
* Response
  + SOAP response for success/failure (SOAP-ENV fault tag)

Processing

The visit\_sid will be check to confirm the record relates to an appointment rather than an empty slot (visit\_slot = true and visit\_type = 1). The outcome fields will be set as per the UHW business logic applied when an appointment outcome is added.

Errors

TBA

## WS51: CreateDiarySlot

This service is to facilitate the creation of slots into the diary, according to the requirements of O.H.G.  
  
Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
  + Data Object: DiarySlotDto
    - (see Appendix A)
* Response
  + SOAP response for success/failure (SOAP-ENV fault tag)

Processing

A slot will be created as per the logic on the UHT diary maintenance forms business logic as long as the day is not marked as “Non-bookable”

Fields set:

Visit\_sid

Visitor\_sid

Visit\_comment

Cisit\_type

Visit\_slot

Visit\_prop\_appointment

Visit\_prop\_end

Visit\_prop\_duration

Hadiary\_sid

Visitor\_table

Visit\_slot\_type

For new slots the system sets visit\_slot to true and visit\_type to 3. Visit\_prop\_end is system calculated from the start datetime and duration values.

Errors

TBA

## WS53: DeleteDiarySlot

This service is to facilitate the deletion of a slot from the diary, according to the requirements of O.H.G.

Webservice

* Data Input Requirements
  + Authentication Elements
    - Master Webservices Password
    - Webservice User ID
    - User Session Token
  + Source System Reference
    - External Source System
    - External System Unique Record ID / Key
  + Visit\_Sid
* Response
  + SOAP response for success/failure (SOAP-ENV fault tag)

Processing

The visit sid will be checked that it relates to a slot record and that it has not been booked before deleting the slot. A free slot is defined as having visit\_slot set to true and visit\_type set to 3.

Errors

TBA

# Appendix A – DTO Object Structures

## NewRepairDto

<NewRepairDto>

<RepairRequestDto>

<See 9.8>

<RepairRequestDto>

*collection of tasks:*

*<Tasks>*

*<RepairTaskDto>*

<see 9.7>

<RepairTaskDto>

</Tasks>

<UserFields>

<KeyValueOfStringString>

<Key>u\_myuserfield</Key>

<Value> ValueToSet</Value>

</KeyValueOfStringString>

</UserFields>

</NewRepairDto>

## RepairTaskDto

<RepairTaskDto>

<Reference></Reference>

<WorksOrderReference></WorksOrderReference>

<TaskNo></TaskNo>

<RepairRequestReference></RepairRequestReference>

<SourceTaskReference></SourceTaskReference>

<JobCode></JobCode>

<TradeReference></TradeReference>

<SupplierReference></SupplierReference>

<TypeCode></TypeCode>

<ClassCode></ClassCode>

<Details></Details>

<DateReported></DateReported>

<DateCreated></DateCreated>

<IssueByDate></IssueByDate>

<DateIssued></DateIssued>

<DateDue></DateDue>

<UserCode></UserCode>

<DateCompleted></DateCompleted>

<PropertyReference></PropertyReference>

<ActualCost></ActualCost>

<EstimatedCost></EstimatedCost>

<EstimatedUnits></EstimatedUnits>

<UnitNarrative></UnitNarrative>

<SorContract></SorContract>

<PriorityReference></PriorityReference>

<InvoiceNo></InvoiceNo>

<DateInvoiced></DateInvoiced>

<TaskStatusCode></TaskStatusCode>

<AttributeReference></AttributeReference>

<AttributeEffectCode></AttributeEffectCode>

<ChildTaskReference></ChildTaskReference>

<IsInstructed></IsInstructed>

<IsAutoGenerated></IsAutoGenerated>

<IsComplete></IsComplete>

<IsAuthorised></IsAuthorised>

<AuthByUserCode></AuthByUserCode>

<DateAuthorised></DateAuthorised>

<IsSuspended></IsSuspended>

<EstimatedFactorCode></EstimatedFactorCode>

<RepairAreaCode></RepairAreaCode>

<UserFields>

<KeyValueOfStringString>

<Key>u\_myuserfield</Key>

<Value> ValueToSet</Value>

</KeyValueOfStringString>

</UserFields>

</RepairTaskDto>

## RepairRequestDto

<RepairRequestDto>

<Reference>

<Date>

<Location>

<UserCode>

<PropertyReference>

<HouseholdReference>

<PersonNo>

<ContactName>

<ContactPhone>

<Details>

<IsOpen>

<TaskReference>

<AccessDetails>

<AccessMonAm>

<AccessMonPm>

<AccessMonSchoolRun>

<AccessTueAm>

<AccessTuePm>

<AccessTueSchoolRun>

<AccessWedAm>

<AccessWedPm>

<AccessWedSchoolRun>

<AccessThuAm>

<AccessThuPm>

<AccessThuSchoolRun>

<AccessFriAm>

<AccessFriPm>

<AccessFriSchoolRun>

<AccessSatAm>

<AccessSatPm>

<AccessSunAm>

<AccessSunPm>

<Comments>

<CancellationCode>

<AppointmentDate>

<AppointmentTime>

<Id>

<IsPreInspection>

<InformedSupplierReference>

<CallContractorInformed>

<CallerTypeCode>

<TypeCode>

<NoDeferrals>

<DateDeferredTo>

<DateDue>

<PriorityReference>

<StatusCode>

<PreInspectionTaskReference>

<WardenReference>

<Location2>

<Location2Details>

<Version>

<OverallStatusCode>

<OverallStatusDate>

<CancellationDate>

<CancellingUserCode>

<CompanyCode>

<IsHazardSuspected>

<VoidMonitoringReference>

<NotifiedDate>

<UserFields>

<KeyValueOfStringString>

<Key>u\_myuserfield</Key>

<Value> ValueToSet</Value>

</KeyValueOfStringString>

</UserFields>

<RepairRequestDto>

## InspectionOutcomeDto

<InspectionOutcomeDto>

<CompletionDate></Completion\_date>

<InspectionOutcomeCode></InspectionOutcomeCode>

<Comments></Comments>

<CancelledReasonCode></CancelledReasonCode>

<DefectiveReasonCode></DefectiveReasonCode>

<UserFields>

<KeyValueOfStringString>

<Key>u\_myuserfield</Key>

<Value> ValueToSet</Value>

</KeyValueOfStringString>

</UserFields>

<InspectionOutcomeDto>

## DiarySlotDto

<DiarySlotDto>

<VisitId></VisitId>

<VisitorId></VisitorId>

<Comment></Comment>

<AppointmentStart></AppointmentStart>

<AppointmentDuration></AppointmentDuration>

<DiaryId></DiaryId>

<VisitorTable></VisitorTable>

<UserFields>

<KeyValueOfStringString>

<Key>u\_myuserfield</Key>

<Value> ValueToSet</Value>

</KeyValueOfStringString>

</UserFields>

</DiarySlotDto>

## VisitOutcomeDto

<VisitOutcomeDto>

<VisitId></VisitId>

<VisitOutcomeCode></VisitOutcomeCode>

<VisitOutcomeTypeCode></VisitOutcomeTypeCode>

<UserFields>

<KeyValueOfStringString>

<Key>u\_myuserfield</Key>

<Value> ValueToSet</Value>

</KeyValueOfStringString>

</UserFields>

</VisitOutcomeDto>

## DiaryAppointmentDto

<DiaryAppointmentDto>

<VisitId> </VisitId>

<ReferenceId></ReferenceId>

<Comment></Comment>

<PropertyId></PropertyId>

<UserFields>

<KeyValueOfStringString>

<Key>u\_myuserfield</Key>

<Value> ValueToSet</Value>

</KeyValueOfStringString>

</UserFields>

<DiaryAppointmentDto>