RAPTOR Data Layer Functionality

Last Updated 20140315

Contents

[Fundamental Data Concepts and Terms in RAPTOR 3](#_Toc382647118)

[Unique Tracking ID 3](#_Toc382647119)

[Application Data Layer 3](#_Toc382647120)

[Ticket Workflow State Codes 3](#_Toc382647121)

[Worklist Modes 3](#_Toc382647122)

[Ticket Tracking 4](#_Toc382647123)

[Class TicketTrackingData 4](#_Toc382647124)

[Constructor 4](#_Toc382647125)

[POC Background 4](#_Toc382647126)

[Function getTrackingID(parameters tbd) 4](#_Toc382647127)

[Function getTicketWorkflowState($sTrackingID) 4](#_Toc382647128)

[Function markTicketLocked($sTrackingID, $sUserID, $sStateCode) 4](#_Toc382647129)

[Function markTicketUnlocked($sTrackingID) 5](#_Toc382647130)

[Function setTicketWorkflowState($sTrackingID, $sStateCode) 5](#_Toc382647131)

[Worklist Data 5](#_Toc382647132)

[Class WorklistData 5](#_Toc382647133)

[Constructor 5](#_Toc382647134)

[POC Background 6](#_Toc382647135)

[Function getWorklistModeName($sWMODE) 6](#_Toc382647136)

[Function getWorklistRows($oContext) 6](#_Toc382647137)

[Function getOneWorklistRow($sTrackingID) 7](#_Toc382647138)

[Protocol Data 7](#_Toc382647139)

[POC Background 8](#_Toc382647140)

[Class DashboardData 8](#_Toc382647141)

[Constructor 8](#_Toc382647142)

[Function getDashboardDetails($sTrackingID) 9](#_Toc382647143)

[Dashboard Associative Array Labels 9](#_Toc382647144)

[Class ProtocolSupportingData 10](#_Toc382647145)

[Constructor 10](#_Toc382647146)

[Function getOrderOverview() 10](#_Toc382647147)

[Function getMedicationsDetail() 10](#_Toc382647148)

[Function getVitalsSummary() 10](#_Toc382647149)

[Function getVitalsDetail() 11](#_Toc382647150)

[Function getAllergiesDetail() 11](#_Toc382647151)

[Function getProcedureLabsDetail() 11](#_Toc382647152)

[Function getDiagnosticLabsDetail() 12](#_Toc382647153)

[Function getDoseHxDetail() 12](#_Toc382647154)

[Function getPathologyReportsDetail() 12](#_Toc382647155)

[Function getSurgeryReportsDetail() 12](#_Toc382647156)

[Function getProblemsListDetail() 12](#_Toc382647157)

[Function getNotesDetail() 13](#_Toc382647158)

[Function getRadiologyReportsDetail() 13](#_Toc382647159)

[Miscellaneous Helper Classes 13](#_Toc382647160)

[Class Context 13](#_Toc382647161)

[Global Variables 13](#_Toc382647162)

[Revision History 15](#_Toc382647163)

**Tables**

[Table 1 - RAPTOR Ticket States 3](#_Toc382647164)

[Table 2 - Worklist Modes 4](#_Toc382647165)

[Table 3 - Labeled Items of the WorklistRows Array 6](#_Toc382647166)

[Table 4 – Values in the Worklist DataRows Array 7](#_Toc382647167)

[Table 5 - Tabs of the Protocol Page 8](#_Toc382647168)

[Table 6 – Dashboard Content 9](#_Toc382647169)

[Table 7 - Dashboard Associative Array Labels 9](#_Toc382647170)

[Table 8 - Global Variables 13](#_Toc382647171)

# Fundamental Data Concepts and Terms in RAPTOR

RAPTOR is primarily a tool for protocoling imaging studies. The following terms are sometimes used interchangeably in this context: study, order, procedure. For purposes of this document, we will use the term “ticket” to describe a study/order/procedure that is being handled by RAPTOR.

## Unique Tracking ID

Each ticket in RAPTOR has to have a unique “Tracking ID”. This tracking id is assigned within RAPTOR so that no two study/order/procedure (“ticket”) ever have the same number and the number always maps back to the original study/order/procedure (“ticket”).

The ticket tracking number is used within RAPTOR to associate protocol and workflow details with the original study/order/procedure. This association exists only within the RAPTOR database.

## Application Data Layer

The RAPTOR application interacts with all external systems and its local database via a data layer of custom PHP classes and functions. Those classes and functions are described in this document. All classes are expected to be part of the “**raptor**” namespace.

NOTE: If a function returns a single dimension array of values, that array should be an associative array unless otherwise stated. This will ensure a greater level of assurance that the values are being used for their intended purpose in the consuming modules.

### Ticket Workflow State Codes

A ticket can only be in one of the following RAPTOR states. The data layer is expected to work with these codes for filtering purposes.

Table - RAPTOR Ticket States

| State Name | State Code |
| --- | --- |
| Active | AC |
| Approved | AP |
| Collaborative | CO |
| Review | RV |
| Protocol Acknowledged | PA |
| Inactive | IA |
| Exam Complete | EC |

### Worklist Modes

One of the worklist filtering criteria is via the worklist mode. The mode filtering works as described in Table 2. The data layer is expected to work with these codes and support this filtering.

Table - Worklist Modes

| Mode Name | Code | Description |
| --- | --- | --- |
| Protocol | P | Display any tickets that are in any of the following workflow states:   * Active * Collaborative * Review |
| Examination | E | Display any tickets that are in any of the following workflow states:   * Approved * Protocol Acknowledged |
| Interpretation | I | Display any tickets that are in any of the following workflow states:   * Exam Complete |
| QA | Q | Display any tickets that are in any of the following workflow states:   * Exam Complete * Inactive |

# Ticket Tracking

A key requirement of the data layer in RAPTOR is that it can associate RAPTOR data with externally sourced data. Internally each ticket gets a unique tracking ID for that purpose.

## Class TicketTrackingData

This is a utility class that does not map to any particular page and should be coded into a file called **data\_ticket\_tracking.php**.

### Constructor

No constructor is necessary for this class. Methods will have static behavior.

### POC Background

The implementation of the tracking identifier in the POC was not sufficient for production purposes. The concept was that there are a set of values that exist in an externally sourced study/order/procedure that can be consistently converted into the same tracking number each time those inputs are provided.

### Function getTrackingID(parameters tbd)

**IMPORANT NOTE:** The data team will need to identify the appropriate set of parameters to provide to this method so that it can return an immutable tracking identifier. These values must always be available from the worklist level.

### Function getTicketWorkflowState($sTrackingID)

Returns one of the values from Table 1. If no state has been recorded for the ticket, then this function returns the status code for the “Active” state.

*This function will be implemented by the presentation layer team.*

### Function markTicketLocked($sTrackingID, $sUserID, $sStateCode)

Calling this function marks a ticket as being edited by a RAPTOR user. The $sStateCode is optional and if not NULL then the ticket is placed into the indicated state (see Table 1 for values).

*This function will be implemented by the presentation layer team.*

### Function markTicketUnlocked($sTrackingID)

Calling this function clears any lock that might have associated the ticket as being edited by a RAPTOR user.

*This function will be implemented by the presentation layer team.*

### Function setTicketWorkflowState($sTrackingID, $sStateCode)

Calling this function sets a ticket as being in a particular RAPTOR workflow state (see Table 1 for values).

*This function will be implemented by the presentation layer team.*

# Worklist Data

The worklist shows all the available orders for a user to protocol. The orders might be in any state. There is also a request from the VA that they be able to create orders in RAPTOR for protocol purposes.

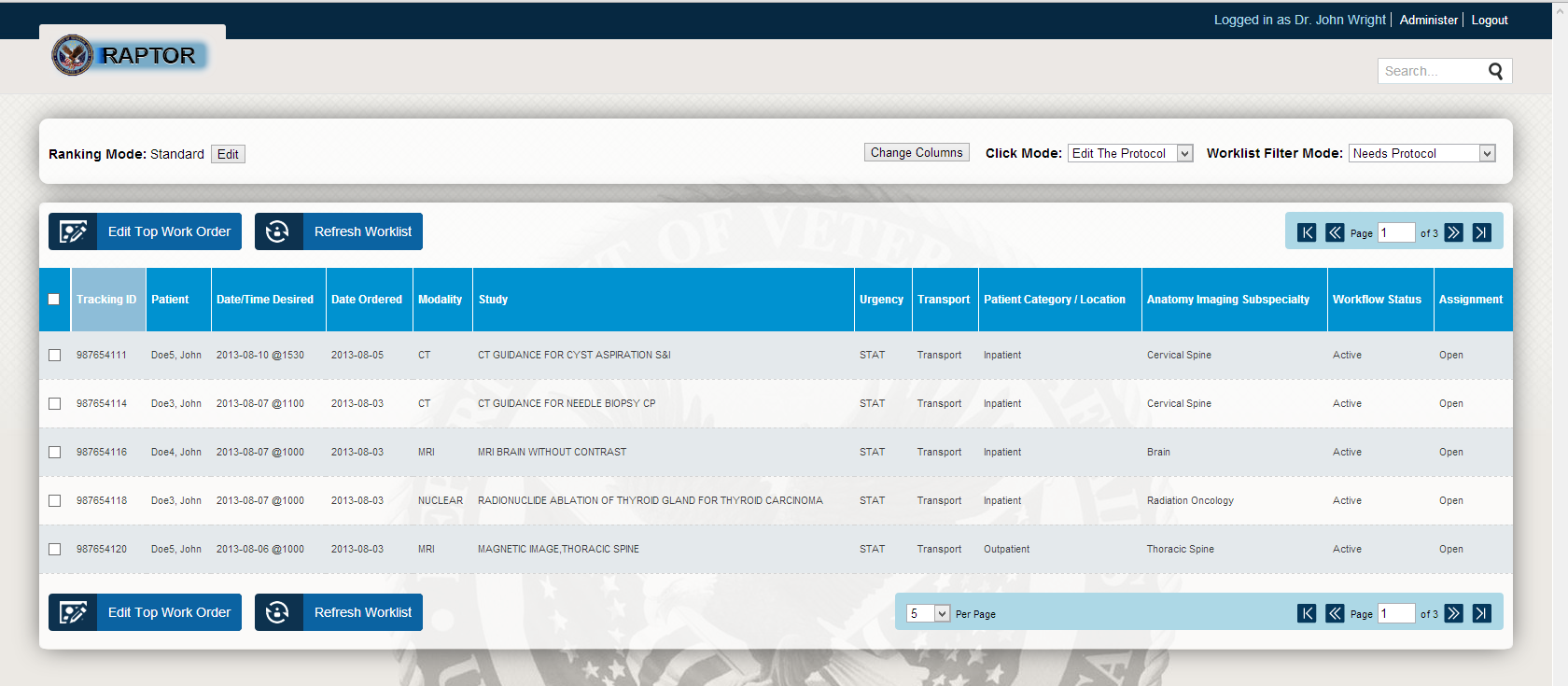


Figure - Sample Worklist Page

The following classes and functions provide the functionality for the application to display and for the user to interact with the worklist data.

## Class WorklistData

This class returns worklist specific data and should be coded into a file called **data\_worklist.php**.

### Constructor

Takes a Context instance as input. The Context is a class managed by the RAPTOR application and contains key values such as UserID of the current user and various filter and sorting options they have selected.

### POC Background

You will find relevant data implementation details in the old **raptor\_worklist.php** and **worklist.php** files. The primary presentation layer function is **\_raptor\_blocks\_get\_worklist\_html** found in the old **page\_content\_html.inc** file.

### Function getWorklistModeName($sWMODE)

This is a static function that returns the word associated with the code.

/\*\*

\* Convert the code into a word

\* @param type $sWMODE

\* @return string the word associated with the code

\*/

static function getWorklistModeName($sWMODE)

{

if($sWMODE=='P'){

$sName="Protocol";

} elseif ($sWMODE=='E'){

$sName="Examination";

} elseif ($sWMODE=='I'){

$sName="Interpretation";

} elseif ($sWMODE=='Q'){

$sName="QA";

} else {

die("Invalid WorklistMode='$sWMODE'!!!");

}

return $sName;

}

### Function getWorklistRows($oContext)

Return a multi-part hybrid associative array of all worklist tuples satisfying the filter criteria implied by the instance of Context class passed in as a parameter.

Table - Labeled Items of the WorklistRows Array

| Item | Label | Value Description |
| --- | --- | --- |
| 1 | Pages | Number of pages found |
| 2 | Page | Current page for which rows are being returned |
| 3 | RowsPerPage | Rows appearing on each page |
| 4 | DataRows | An array of non-associative array data rows. |

Each array data row contains the values shown in Table 4.

Table – Values in the Worklist DataRows Array

| Order | Name | Description |
| --- | --- | --- |
| 1 | Tracking ID | This is a unique ID assigned by the RAPTOR application to the order. At any time, the raptor application can request this order simply by providing this value as a parameter to the data layer functions. |
| 2 | Patient | “Last name, First name” of the patient |
| 3 | Date/Time Desired | The date and time that the procedure is requested to occur |
| 4 | Date Ordered | The date (and time) that the order was placed |
| 5 | Modality | The modality of the procedure |
| 6 | Study | Description of the ordered procedure |
| 7 | Urgency | One of several values: STAT, URGENT, *blank* |
| 8 | Transport | One of several keywords |
| 9 | Patient Category / Location | One of several keywords |
| 10 | Anatomy Image Subspecialty | One or more anatomy keywords |
| 11 | Workflow Status | The current RAPTOR workflow state of this ticket |
| 12 | Assignment | Name of assigned RAPTOR user if any has been assigned for collaboration or review. If no one has been assigned, then this value is blank. |
| 13 | Order Status | The status keyword from external system (same value as seen in CPRS) |
| 14 | Current User ID | If the ticket is currently being edited by a RAPTOR user, this is the User ID of that user. Otherwise this is blank. |

The presentation layer sorts and filters the worklist array such that only a subset of the rows and columns are displayed.

**Filtering Note:** If an order/procedure/study is ACTIVE or PENDING in CPRS and there is no existing Tracking ID already associated with it in RAPTOR, then for RAPTOR purposes that order/study/procedure is assumed to be in the RAPTOR “Active” state.

### Function getOneWorklistRow($sTrackingID)

Return all the values for only one worklist tuple as described in Table 4. The $sTrackingID uniquely identifies the item.

# Protocol Data

The protocol data context for RAPTOR has a main page in which the user enters/edits protocol information and several sub-tab pages in which they primarily view supporting detail information.

Table - Tabs of the Protocol Page

| Order | Tab Name | Description |
| --- | --- | --- |
| 1 | Protocol | This is the default selected tab when a user lands on the Protocol Page. This tab area is where all the protocol settings are captured and some read-only background information is made available to help with the protocol process. This is also the tab area where a technician enters notes during the procedure. |
| 2 | Medications | Displays medication details |
| 3 | Vitals | Displays vitals details |
| 4 | Allergies | Displays allergy details |
| 5 | Labs | Displays lab details depending on the type of ticket |
| 6 | Dose Hx | Displays dose details |
| 7 | Clinical Reports | Displays clinical report details |
| 8 | Problems List | Displays problem details |
| 9 | Notes | Displays note details |
| 10 | Radiology Reports | Displays radiology report details |
| 11 | *Protocol Library* | *This is where we find a listing of all the existing protocols available at the hospital center*  ***NOTE: The presentation layer team will implement these functions.*** |

## POC Background

The content for the pages in the POC are primarily produced by the **\_raptor\_blocks\_get\_detail\_html** function of the old **page\_content\_html.inc** file. You can navigate back from that point to see how the data is extracted from MDWS in the POC.

## Class DashboardData

This class provides the dashboard (formerly known as “boilerplate” in some documents) area content data. This data is a presentation of key information specific to the selected ticket. The look and feel of the dashboard is illustrated in Figure 2. The presentation layer handles all the formatting of the data.

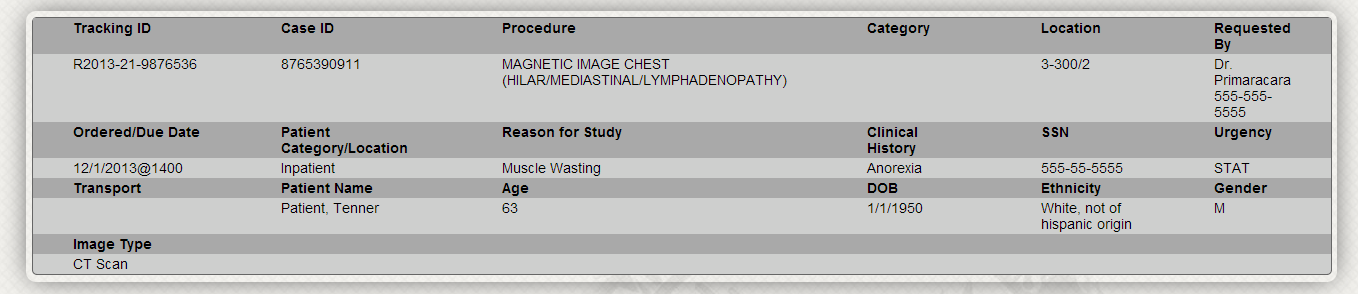


Figure - Look and feel of Dashboard Area

The implementation of this class is found in the **data\_dashboard.php** file.

### Constructor

This class needs no constructor because the methods have static behavior.

### Function getDashboardDetails($sTrackingID)

This function returns one tuple with the values described in Table 6. The tuple should be an instance of an associative array with each value label reasonably matching the label name show in Table 6.

Table – Dashboard Content

|  | Dashboard Label | Expected Max Characters | Description |
| --- | --- | --- | --- |
| 1 | Tracking ID | 50 | RAPTOR unique tracking ID for the procedure. RAPTOR must create this number because it stores data locally. |
| 2 | Case ID | 50 | From VistA Case Name field |
| 3 | Procedure | 250 | From VistA Rad Procedure field. |
| 4 | Category | 50 | From VistA Category field |
| 5 | Location | 50 | Room/Bed for inpatients |
| 6 | Requested By | 100 | Name and Phone# of requesting physician |
| 7 | Requested Date | 10 | The MM/DD/YYYY date on which the order was placed. |
| 8 | Scheduled Date | 10 | The MM/DD/YYYY date on which the procedure is scheduled to take place |
| 9 | Patient Category | 12 | ???? inpatient vs outpatient ??? |
| 10 | Reason for Study | 250 | From VistA Reason field |
| 11 | Clinical History | 250 | From VistA Clinical History field |
| 12 | SSN | 11 | From VistA Patient SSN field (###-##-####) |
| 13 | Urgency | 20 | From VistA Urgency field |
| 14 | Transport | 50 | From VistA Transport field |
| 15 | Patient Name | 80 | From VistA Patient Name field. Display as ***Last, First***. |
| 16 | Age | 3 | From VistA Age field |
| 17 | DOB | 10 | From VistA DOB field. Display as MM/DD/YYYY |
| 18 | Ethnicity | 10 | From VistA Ethnicity field. |
| 19 | Gender | 1 | From VistA Gender field. Display as M or F. |
| 20 | Image Type | 20 | From VistA Image Type field |

### Dashboard Associative Array Labels

The instance of the dashboard is returned as an associative array. The following snippet illustrates the syntax for an associate array that contains only 2 of the required values.

return Array(“TrackingID”=>$sTrackingID, “CaseID” => $sCaseID);

The expected labels for the associative array are described in Table 7.

Table - Dashboard Associative Array Labels

|  | Dashboard Label | Array Label | Value Description |
| --- | --- | --- | --- |
| 1 | Tracking ID | TrackingID | Contains string value |
| 2 | Case ID | CaseID | Contains string value |
| 3 | Procedure | Procedure | Contains string value |
| 4 | Category | Category | Contains string value |
| 5 | Location | PatientLocation | Contains string value |
| 6 | Requested By | RequestedBy | Contains string value |
| 7 | Requested Date | RequestedDate | Contains string value **MM/DD/YYYY** |
| 8 | Scheduled Date | ScheduledDate | Contains string value **MM/DD/YYYY @ HH:MM** |
| 9 | Patient Category | PatientCategory | Contains string value |
| 10 | Reason for Study | ReasonForStudy | Contains string value |
| 11 | Clinical History | ClinicalHistory | Contains string value |
| 12 | SSN | PatientSSN | Contains string value **###-##-####** |
| 13 | Urgency | Urgency | Contains string value |
| 14 | Transport | Transport | Contains string value |
| 15 | Patient Name | PatientName | Contains string value **Last, First** |
| 16 | Age | PatientAge | Contains string value |
| 17 | DOB | PatientDOB | Contains string value **MM/DD/YYYY** |
| 18 | Ethnicity | PatientEthnicity | Contains string value |
| 19 | Gender | PatientGender | Contains string value |
| 20 | Image Type | ImageType | Contains string value |

## Class ProtocolSupportingData

This class provides the data that is displayed in the static information area of the main protocol tab and also the detailed content for the specialized tabs.

### Constructor

This constructor should take an instance of the ***Context*** class.

### Function getOrderOverview()

Return a single associative array with the following values for the ticket.

1. Requested By
2. Primary Care Physician
3. Attending Physician
4. Requested Study
5. Reason for Study

### Function getMedicationsDetail()

Return an associative array of all data to display. Each row of array should contain the following data:

1. Medication
2. At Risk
3. Status

### Function getVitalsSummary()

Return an associative array of all data to display. There is only one row of key value pairs and the keys are the labels meant for display to the user.

1. Temperature
2. Heart Rate
3. Blood Pressure
4. Height
5. Weight
6. Body Mass Index

Each of the keys above is associated with an array with the following content.

1. Date of Measurement
2. Measurement value

Programmer Note: Do not look for the keys listed here. Instead simply use them as labels. Additional keys and their values may be added in the future and we would like them to simply flow through to the display.

### Function getVitalsDetail()

Return an associative array of all data to display. Each row of array should contain the following data:

1. Date Taken
2. Temp
3. Height
4. Weight
5. BMI
6. Blood Pressure
7. Pulse
8. Resp
9. Pain
10. C/G
11. Pox
12. CVP
13. Blood Glucose

### Function getAllergiesDetail()

Return an associative array of all data to display. Each row of array should contain the following data:

1. Date
2. Item
3. Causative Agent
4. Signs/Symptoms
5. Drug classes
6. Originator
7. Verified
8. Observed/ Historical

### Function getProcedureLabsDetail()

Return an associative array of all data to display. Each row of array should contain the following data:

1. Date
2. PLT
3. PT
4. INR
5. PTT
6. HCT
7. Ref

### Function getDiagnosticLabsDetail()

Return an associative array of all data to display. Each row of array should contain the following data:

1. Date
2. Creatinine
3. eGFR
4. Ref

### Function getDoseHxDetail()

Return an associative array of all data to display. Each row of array should contain the following data:

1. Action
2. Exam Date
3. Procedure Type
4. Height
5. Weight
6. BMI
7. CTDIvol (mGy)
8. DLP (mGy cm)
9. Radiotracer Dose (mCi)
10. Dose Type
11. Entry State

### Function getPathologyReportsDetail()

Return an associative array of all data to display. Each row of array should contain the following data:

1. Title
2. Date
3. Details

### Function getSurgeryReportsDetail()

Return an associative array of all data to display. Each row of array should contain the following data:

1. Title
2. Date
3. Details

### Function getProblemsListDetail()

Return an associative array of all data to display. Each row of array should contain the following data:

1. Title
2. Onset Date
3. Details

### Function getNotesDetail()

Return an associative array of all data to display. Each row of array should contain the following data:

1. Type
2. Date
3. Details

### Function getRadiologyReportsDetail()

Return an associative array of all data to display. Each row of array should contain the following data:

1. Title
2. Date
3. Details

# Miscellaneous Helper Classes

*These classes are implemented by the presentation layer team.*

## Class Context

This class is a singleton instance for the application session and describes key information for the user’s session.

* Function getUserID()
  + Returns the RAPTOR User ID of the currently logged in user.
* Function getWorklistMode()
  + Returns the currently selected worklist mode for the session. The value is a code as shown in Table 2.
* Function getSelectedTrackingID()
  + Returns the tracking ID of the currently selected ticket or NULL if none is selected

# Global Variables

The **raptor\_omega** theme gets relevant values via the global variables shown in Table 8Table 1. They are only populated as needed by the application context.

Table - Global Variables

| Variable Name | Value Description |
| --- | --- |
| $raptor\_worklist\_rows | Result of getWorklistRows call |
| $raptor\_protocoldashboard | Result of getDashboardDetails call |
| $raptor\_protocol\_content | Contains arrays of arrays for the protocol page and its tabs. There are three top level named sections in the array as follows...   1. Reference 2. ContraIndications 3. Input   Named contents inside of section called "**Reference**" are as follows…   * ["**OrderOverview**"] has getOrderOverview() * ["**VitalsSummary**"] has getVitalsSummary() * ["**MedicationsDetail**"] has getMedicationsDetail() * ["**VitalsDetail**"] has getVitalsDetail() * ["**AllergiesDetail**"] has getAllergiesDetail() * ["**ProcedureLabsDetail**"] has getProcedureLabsDetail() * ["**DiagnosticLabsDetail**"] has getDiagnosticLabsDetail() * ["**DoseHxDetail**"] has getDoseHxDetail() * ["**PathologyReportsDetail**"] has getPathologyReportsDetail() * ["**ReportsDetail**"] has getSurgeryReportsDetail() * ["**ProblemsListDetail**"] has getProblemsListDetail() * ["**NotesDetail**"] has getNotesDetail() * ["**RadiologyReportsDetail**"] has getRadiologyReportsDetail()   Named contents inside of section called “**ContraIndications**” are as follows…   * TBD   Named contents inside of section called “**Input**” are as follows…   * TBD |
|  |  |

# Revision History

| When | Who | Description |
| --- | --- | --- |
| 20140111 | Frank Font | Initial draft based on content from POC and architecture of phase II RAPTOR application. |
| 20140217 | Frank Font | Added mention of “raptor” namespace and expectation that values are returned in associative arrays. Renamed boilerplate to dashboard for clarity. Also split requested date and scheduled date in the dashboard. Also added getStaticModeName function. Simplified Dashboard Content. Created section for global variables. |
| 20140222 | Frank Font | Additional global variable |
| 20140301 | Frank Font | Added getVitalsSummary |
| 20140315 | Frank Font | Removed getTicketType based on SME discussion last week that diagnostic/image guided criteria will be resolved via protocol and contra indication settings rather than derived from order information. |