COLORADO HEALTHTRAC

CODA

Clinical Outcomes Decision Architecture

Developer Guide

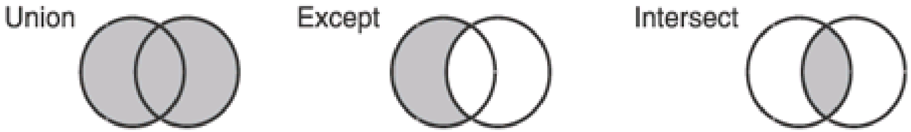
KAISER PERMANENTE®

Revision History

Version	Date	Author	Description of Revision
1.0	4-25-2013	Pete Champlin	Document creation
[redacted]	[redacted]	[redacted]	[redacted]
1.2	12-09-2013	Pete Champlin	Add Gap Due Date section.
1.3	2-14-2014	Pete Champlin	Add Gap Solution and modify Rule information. Add development guidelines.
1.4	5-19-2014	Pete Champlin	Add due date and gap reason calculation information.
[redacted]	[redacted]	[redacted]	[redacted]

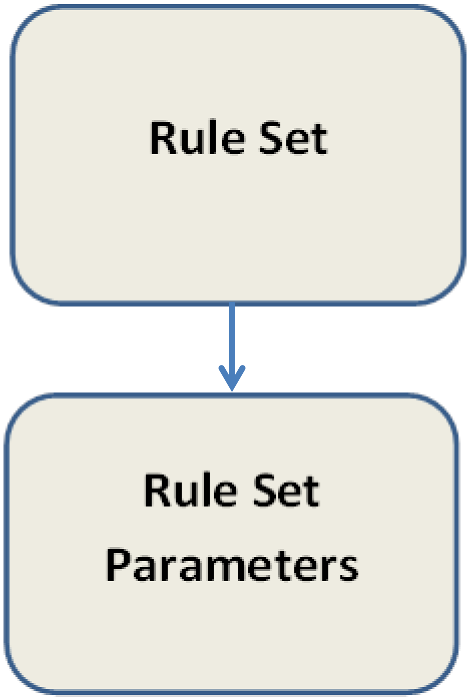
What is CODA and how does it work?

- A relational database solution to determine a set of members who meet a series of conditions
- Takes advantage of database set processing optimizations
- Conditionally outputs the selected members to various destination tables
- Rule logic is contained in parameterized SQL stored procedures
- Includes versioned, historical objects
- Provides visibility into the “reasons” for members’ inclusion in or exclusion from the outcome
- Allows gaps to be pre-calculated and persisted
- All objects are stored in a separate CODA database
- Uses the SQL Set Operators - INTERSECT, UNION, and EXCEPT - applied in a sequence, to combine and filter member sets:



Rule Sets

CODA is based on Rule Sets, which are subsets of members who meet the criteria contained in the Rule Set definition. Each Rule Set references a “rule set” stored procedure, which contains any logic necessary to select specific members. Rule Set stored procedure can also take required or optional parameters, allowing the procedure to be flexibly reused in multiple scenarios.



Rule Set

- Defines a subset of members who meet a specific condition
- Specifies a Rule Set Stored Procedure
- Reusable in multiple rules
- Stored in the RuleSet table

Rule Set Parameters

- Specifies the Rule Set condition
- Includes parameter name(s) and value(s)
- Allows Rule Set Stored Procedures to be used for multiple Rule Sets
- Stored in the RuleSetParam and RuleSetParamValue tables

Creating a Rule Set

A general development guideline is to make Rule Sets quite specific, so that it’s readily apparent why members are or are not selected into the Rule Set. For example, if a Rule Set is designed to include members who have recent results for a certain Lab Component group and are on a certain medication, it would then be more difficult to determine why members were not included in the set (i.e. did they not have a recent lab result or were they not on the medication?). It would be better to create two Rule Sets – one for Lab Component and one for the medication – and then intersect them during the processing of the Rule.

Creating a RuleSet Stored Procedure

The purpose of RuleSet stored procedures is to select members who meet desired criteria. Any logic needed to accomplish this can be included in the procedure, which will write the specified MemberIds to the MemberRuleSet table, along with a RuleSetId. Before the MemberIds are written to the table, existing rows for the specified RuleSetId are first deleted, so the MemberRuleSet table will always contain the set of members who meet the Rule Set's criteria as of the last time the Rule Set was processed.

Rule Set stored procedures always take a required @RuleSetId parameter and an optional @MemberId parameter (which allows a Rule Set to be processed for just one member), but can also have any other required or optional parameters that could make the procedure reusable in multiple scenarios. For example, the procedure usp_RuleSet_MedicationGroup takes a required @MedicationGroup parameter, which allows members in the specified medication group to be selected based on the parameter value. So, this one procedure can be used for any Rule Set designed to determine whether members are currently on a medication of a specified medication group.

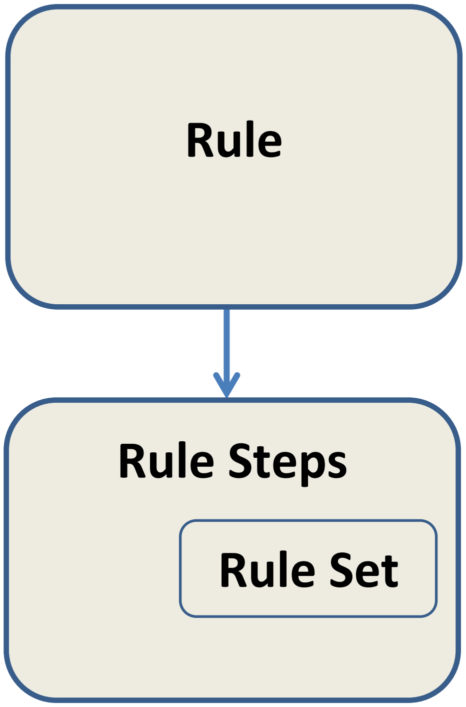
Many Rule Set stored procedures already exist, so before developing a new one always check whether there is a procedure that meets your needs.

Rule Set stored procedure names are prefixed with "usp_RuleSet" and always contain code which first deletes from and then inserts into the MemberRuleSet table.

Rules & Rule Steps

CODA Rules are a collection of Rule Sets that are processed in a specified sequence. The Rule Steps associated with a Rule define the sequence in which the Rule Sets are processed and how each Rule Set is combined or compared with Rule Sets that are processed either before or after it. The result is a final set of members that meet the Rule's criteria.

The Rule Type attribute is used to direct the output member set to specified tables. For example, "Gap" type rules will cause output members to be written to the MemberGap table, while "AuditRelease" type rules may write to the AuditRelease tables.



- A collection of Rule Sets
- Includes name and description
- Associated Rule Type determines processing and storage of output
- Versioned
- Stored in the RuleDefinition table

- Defines order and relation of Rule Sets within a rule
- Includes:
  - Sequence
  - Set Operator (Union, Intersect, Except)
  - Open and close parentheses (optional)
- Versioned
- Stored in the RuleStep table

Troubleshooting Rules

Once the Rule Sets, Rule, and Rule Steps are created, there are a number of procedures that can be run to review those records:

- The procedure usp_GetMemberRuleSet can be run with a @Debug parameter value equal to 1, which will simply print the SQL that will be run to populate the MemberRuleSet table. This can be a good way to review the parameters that will be passed to each Rule Set stored procedure referenced in the Rule.

- The procedure usp_ReviewRuleByMember is normally run with a @MemberId parameter in order to review the reasons why a member was or was not included in the final member output set, but it can be run without that parameter in order to review the Rule Step logic. The Set Operators INTERSECT, UNION, and EXCEPT will be translated to "And", "Or", and "Not," respectively.

  ```
  EXEC coda.usp_ReviewRuleByMember @RuleId = 6
  ```

(On a thyroid medication
Or Graves Disease
Or Hypothyroid
Or Hyperthyroid)
And Last TSH > 10 Months

After the Rule is processed (see section below) you can run this procedure for the Rule and a member in order to review the outcomes for that member:

```
EXEC coda.usp_ReviewRuleByMember @RuleId = 6, @MemberId = 1564
```

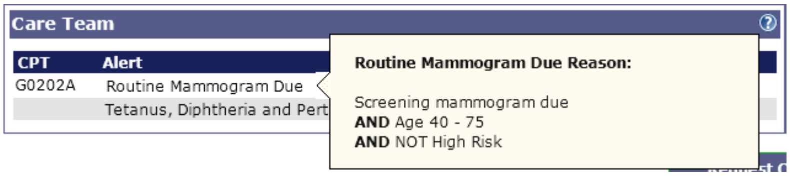
RuleStep	Outcome
(On a thyroid medication	1
Or Graves Disease	0
Or Hypothyroid	1
Or Hyperthyroid)	0
And Last TSH > 10 Months	1

You can see that this member is on a Thyroid medication and in the Hypothyroid Filter and she has not had a TSH lab test in over 10 months.

Gap Features

A primary function of CODA is to calculate care gaps for members. Any "gap" type Rule will cause data to be written to the CODA MemberGap table; new gaps will be created and existing gaps will be closed as needed. Because care gaps are so central to the mission of the HealthTRAC application, the following robust gap-related features have been developed:

- **Gap Reason** – Hovering over a gap on the HealthTRAC Landing Page will display one or more gap reasons, which are calculated at the time gap Rules are processed.



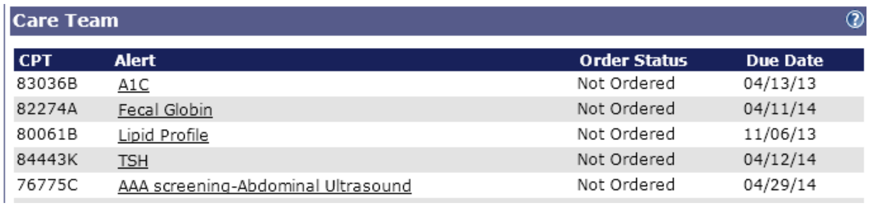
Care Team

CPT	Alert
G0202A	Routine Mammogram Due
	Tetanus, Diphtheria and Pert

Routine Mammogram Due Reason:

Screening mammogram due
AND Age 40 - 75
AND NOT High Risk

- **Due Date** – The Landing Page also displays a due date for each Care Team CODA gap. Due dates are calculated each time gap Rules are processed.



Care Team

CPT	Alert	Order Status	Due Date
83036B	A1C	Not Ordered	04/13/13
82274A	Fecal Globin	Not Ordered	04/11/14
80061B	Lipid Profile	Not Ordered	11/06/13
84443K	TSH	Not Ordered	04/12/14
76775C	AAA screening-Abdominal Ultrasound	Not Ordered	04/29/14

- **Gap Solution** – Actions that can be taken in response to a gap and its due date, for example, displaying the gap on the Landing Page or auto ordering procedures prior to the due date, or notifying members by sending letters or emails either before or after the due date.

Potential Gaps

A potential gap is a gap record that is created when a member meets all the criteria of the Rule (or of at least one Rule Group of the Rule) except for any Rule Step flagged as a “Due Date” step. The RuleStep table contains a column called IsDueDateStep, indicating that the Rule Step contains gap interval information, and can be used to calculate a due date. If a member satisfies all of the Steps of a Rule’s Rule Group, then the member will have a current gap; however, if all of the Steps of a Rule’s Rule Group are satisfied except for any Due Date steps, then the gap row will still be created, but as a potential gap, with a due date in the future. If all conditions remain the same, then the gap will become due after the passage of a certain amount of time. For example, even if a member has a recent TSH lab result, she will still receive a TSH (potential) gap record if she is on a thyroid medication or in the Graves Disease, Hypothyroid, or Hyperthyroid filter.

When the gap Rule is processed, the SQL that is generated to select the members who have the gap will be modified to exclude the Due Date Step. For TSH, the final Step, “Last TSH > 10 Months”, would be flagged as a Due Date Step, and the selection rule would, in effect, change to be:

```
 ( On a thyroid medication
Or  Graves Disease
Or  Hypothyroid
Or  Hyperthyroid )
```

Notice that the final Step is not included, so members who meet at least one of these criteria would receive a TSH gap record in the MemberGap table, regardless of when their most recent TSH lab was. So, the members who have open gap records in the MemberGap table will include those with a “current” gap - with the due date of today or in the past - and those with a future (or potential) gap -with the due date in the future.

CODA Development Guidelines

Below are common questions that must be answered either during development or perhaps even during the requirements-gathering and grooming processes:

- **Creating Rule Sets**
    - Is there already a Rule Set that can be used?
    - Is there already a Rule Set stored procedure that can be used?
    - If creating a new Rule Set stored procedure
        - Do the parameters allow flexible reuse of the procedure?
        - Is it possible to use existing MemberFactor data?
        - Is additional MemberFactor data needed?
        - Would the additional MemberFactor data possibly be used for other Rule Sets?
    - Have the configured stored procedure and parameters been verified?
- **Creating Rules**
    - What is the Rule's Type?
    - If it's a new Type
        - Is there a ProcessRule_<Type> procedure?
        - Has the procedure usp_ProcessRule been modified to call that procedure?
        - Are there destination tables for the output members?
    - If it's a gap Rule
        - Which gap solutions will be triggered?
            - Is there an Action Code and CPT code associated to the gap?
            - Are the gap solution tables configured?
        - Is a calculated due date (and thus a DueDateProcedure) required?
            - Can the due date be calculated using the default or standard model?
        - If converting an existing non-CODA gap, has parallel testing been done?
    - Is a Rule reason needed?
        - Has the text of the reason for display been confirmed?
        - Are Rule Groups configured?
- **Creating Rule Steps**
    - Are the required Rule Sets already created?
    - Does the parentheses and set operator logic make sense?
    - Has the generated SQL been verified?
- **Rule Processing**
    - How often will the Rule be processed?
    - What procedures and/or jobs will be used to process the Rule and its Rule Sets?
    - Is performance acceptable?