Document Version and Date 27 May 2013

By MishaalGhansar

[mishaal.ghansar@mastek.com](mailto:mishaal.ghansar@mastek.com)

**Distribution List**

|  |  |
| --- | --- |
| Sr. No. | Name |
|  | Creative Virtual Ltd. |
|  | Mastek Ltd. |

AMENDMENT HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| Version No. | Date | Prepared By /Modified By | Nature of Changes |
| 0.1 | 03 April 2011 | MishaalGhansar | Draft |
| 0.2 | 04 April 2011 | MishaalGhansar | Updated related FAQ to add more details based on current business area and channel |
| 0.3 | 05 April 2011 | MishaalGhansar | Updated related FAQs and added operator changes. |
| 0.4 | 07 April 2011 | MishaalGhansar | Updated as per review comments from Olaf and added DTree use case |
| 0.5 | 13 April 2011 | MishaalGhansar | Updated as per review comments from Olaf, also added few more use cases for DTree |
| 0.6 | 17 April 2011 | MishaalGhansar | Updated as per review comment. Detailed and added use cases for Macro Indexing and Index Based Search (section 3).Also added V-engine requirement for Dynamic answers & Table answers. |
| 0.7 | 18April 2011 | MishaalGhansar | Added data logging and use cases –section 6 |
| 0.8 | 21 April 2011 | MishaalGhansar | Corrected section 3.1.5, added NLP process KB structure |
| 0.9 | 28 April 2011 | MishaalGhansar | Updated as per review comments from Olaf and added flowcharts for NLP process section 9 |
| 0.10 | 06 May 2011 | MishaalGhansar | Updated Statistical data based related FAQ and Auto -Suggest |
| 0.11 | 11 May 2011 | MishaalGhansar | Updated as per review comments from Olaf and added section request types, python script events, variable filling & Testing |
| 0.12 | 20 May 2011 | MishaalGhansar | Updated as per review comments |
| 0.13 | 31May 2011 | MishaalGhansar | Updated as per review comments from Olaf and Peter. |
| 0.14 | 29 June 2011 | MishaalGhansar | Updated as per review comments from Olaf and Peter. |
| 0.15 | 11 July 2011 | MishaalGhansar | Updated as per review comments from Olaf and Peter. |
| 0.16 | 27 July 2011 | MishaalGhansar | Added examples for Statistical based RQ as per the spreadsheet. |
| 0.17 | 04 August 2011 | MishaalGhansar | Updated as per review comments from Olaf. |
| 0.18 | 11 August 2011 | MishaalGhansar | Updated as per review comments from Olaf. |
| 0.19 | 30 August 2011 | MishaalGhansar | Updated as per review comments from Olaf. |

Review Panel

|  |  |
| --- | --- |
| **Name** | **Role** |
| Peter Behrend | CTO, Creative Virtual |
| Olaf Voß | Project Manager, Creative Virtual |
|  |  |

Reference Documents

|  |  |  |
| --- | --- | --- |
| **Reference ID** | **Document Name** | **Document Version & Date** |
| [R1] | v-portal engine changes 20110314.doc | v0.3, 14 March 2011 |
| [R2] | v-portal system description\_mastek0104 | V1.3, 15 March 2011 |
| [R3] | Requirement Definition Document | v2.1, 17July-2009 |
| [R4] | Decision Trees Objects 20110328A | V0.4, 18 Feb 2011 |
| [R5] | macro indexing |  |
| [R6] | Statistical Autosuggest\_20110425 | 25 April 2011 |

[1 Scope 11](#_Toc302982654)

[2 Use Case: Related QUESTIONS 11](#_Toc302982655)

[2.1 Use Case: Related Questions – using NLP search continuation 12](#_Toc302982656)

[2.1.1 Actors 12](#_Toc302982657)

[2.1.2 Trigger 12](#_Toc302982658)

[2.1.3 Description 12](#_Toc302982659)

[2.1.4 Preconditions 12](#_Toc302982660)

[2.1.5 Postconditions 12](#_Toc302982661)

[2.1.6 Normal Flow 13](#_Toc302982662)

[2.1.7 Alternative Flow 1- 13](#_Toc302982663)

[2.1.8 Alternative Flow2 - 13](#_Toc302982664)

[2.1.9 Hit Rating Calculation 13](#_Toc302982665)

[2.2 Use Case: Related Questions – Semantic Auto-suggest 14](#_Toc302982666)

[2.2.1 Actors 14](#_Toc302982667)

[2.2.2 Trigger 14](#_Toc302982668)

[2.2.3 Description 14](#_Toc302982669)

[2.2.4 Preconditions 14](#_Toc302982670)

[2.2.5 Postconditions 15](#_Toc302982671)

[2.2.6 Normal Flow 15](#_Toc302982672)

[2.2.7 Alternative Flow - 15](#_Toc302982673)

[2.3 Use Case: Related Questions – using FAQ category structure 18](#_Toc302982674)

[2.3.1 Actors 18](#_Toc302982675)

[2.3.2 Trigger 18](#_Toc302982676)

[2.3.3 Description 18](#_Toc302982677)

[2.3.4 Preconditions 18](#_Toc302982678)

[2.3.5 Postconditions 18](#_Toc302982679)

[2.3.6 Normal Flow 18](#_Toc302982680)

[2.3.7 Alternative Flow1- 19](#_Toc302982681)

[2.3.8 Alternative Flow 2- 19](#_Toc302982682)

[2.3.9 Algorithm to determine FAQs based on category- 19](#_Toc302982683)

[2.4 Related Questions – using Statistical Data 20](#_Toc302982684)

[2.4.1 Use Case: Statistical information calculation and logging. 25](#_Toc302982685)

[2.4.2 Use Case: Statistical information usage for Related FAQs. 25](#_Toc302982686)

[2.5 Use Case: Related FAQ – All the 4 algorithm selected in configuration file 27](#_Toc302982687)

[2.5.1 Actors 27](#_Toc302982688)

[2.5.2 Trigger 27](#_Toc302982689)

[2.5.3 Description 27](#_Toc302982690)

[2.5.4 Preconditions 27](#_Toc302982691)

[2.5.5 Postconditions 27](#_Toc302982692)

[2.5.6 Normal Flow 27](#_Toc302982693)

[3 MacroIndexing&Index-BasedSearch 28](#_Toc302982694)

[3.1 Macro Indexing 28](#_Toc302982695)

[3.1.1 Algorithm for creating the Macro index: 28](#_Toc302982696)

[3.1.2 Use Case: V-Portal publishes the KB with macro index information: 30](#_Toc302982697)

[3.1.3 Use Case: User adds a new alternate in V-Portal 30](#_Toc302982698)

[3.1.4 Use Case: User adds/amends/deletes a macro in V-Portal 32](#_Toc302982699)

[3.1.5 Use Case: Restart V-Engine 34](#_Toc302982700)

[3.1.6 Use Case: Batch mode run to update Macro indexing 34](#_Toc302982701)

[3.2 Algorithm for Index based search (uses Macro indexing): 35](#_Toc302982702)

[4 Operators 38](#_Toc302982703)

[4.1 Immediately Followed by operator = 38](#_Toc302982704)

[4.1.1 Operator evaluation 38](#_Toc302982705)

[4.1.2 Restrictions 38](#_Toc302982706)

[4.1.3 Examples 38](#_Toc302982707)

[4.2 Loose Immediately Followed by operator > 39](#_Toc302982708)

[4.2.1 Operator evaluation 39](#_Toc302982709)

[4.2.2 Restrictions 39](#_Toc302982710)

[4.2.3 Examples 39](#_Toc302982711)

[4.3 Pipe operator with macros and bracketed structures. 40](#_Toc302982712)

[4.3.1 Operator evaluation 40](#_Toc302982713)

[4.3.2 Restrictions 41](#_Toc302982714)

[4.3.3 Examples 41](#_Toc302982715)

[5 Use Case: DTREE 43](#_Toc302982716)

[5.1 Use Case: DTree initialization 46](#_Toc302982717)

[5.1.1 Actors 46](#_Toc302982718)

[5.1.2 Trigger 46](#_Toc302982719)

[5.1.3 Description 46](#_Toc302982720)

[5.1.4 Preconditions 46](#_Toc302982721)

[5.1.5 Postconditions 46](#_Toc302982722)

[5.1.6 Normal Flow 46](#_Toc302982723)

[5.1.7 Alternative Flow- 47](#_Toc302982724)

[5.2 Use Case: DTree Entry through special request. 47](#_Toc302982725)

[5.2.1 Actors 47](#_Toc302982726)

[5.2.2 Trigger 47](#_Toc302982727)

[5.2.3 Description 47](#_Toc302982728)

[5.2.4 Preconditions 48](#_Toc302982729)

[5.2.5 Postconditions 48](#_Toc302982730)

[5.2.6 Normal Flow 48](#_Toc302982731)

[5.2.7 Alternative Flow 1- 48](#_Toc302982732)

[5.2.8 Alternative Flow 2- 48](#_Toc302982733)

[5.3 Use Case: DTree -REPROMPT. 49](#_Toc302982734)

[5.3.1 Actors 49](#_Toc302982735)

[5.3.2 Trigger 49](#_Toc302982736)

[5.3.3 Description 49](#_Toc302982737)

[5.3.4 Preconditions 49](#_Toc302982738)

[5.3.5 Postconditions 49](#_Toc302982739)

[5.3.6 Normal Flow 49](#_Toc302982740)

[5.3.7 Alternative Flow- 50](#_Toc302982741)

[5.4 Use Case: DTree Exit. 50](#_Toc302982742)

[5.4.1 Actors 50](#_Toc302982743)

[5.4.2 Trigger 50](#_Toc302982744)

[5.4.3 Description 50](#_Toc302982745)

[5.4.4 Preconditions 50](#_Toc302982746)

[5.4.5 Postconditions 50](#_Toc302982747)

[5.4.6 Normal Flow 50](#_Toc302982748)

[5.4.7 Alternative Flow 1- 51](#_Toc302982749)

[5.4.8 Alternative Flow 2- 51](#_Toc302982750)

[5.5 Use Case: DTreeTermination 51](#_Toc302982751)

[5.5.1 Actors 51](#_Toc302982752)

[5.5.2 Trigger 51](#_Toc302982753)

[5.5.3 Description 51](#_Toc302982754)

[5.5.4 Preconditions 51](#_Toc302982755)

[5.5.5 Postconditions 51](#_Toc302982756)

[5.5.6 Normal Flow 51](#_Toc302982757)

[5.5.7 Alternative Flow 1- 52](#_Toc302982758)

[5.5.8 Alternative Flow 2- 52](#_Toc302982759)

[5.6 Use Case: DTree Navigation/Continuation –Click Continuation 53](#_Toc302982760)

[5.6.1 Actors 53](#_Toc302982761)

[5.6.2 Trigger 53](#_Toc302982762)

[5.6.3 Description 53](#_Toc302982763)

[5.6.4 Preconditions 53](#_Toc302982764)

[5.6.5 Postconditions 53](#_Toc302982765)

[5.6.6 Normal Flow 53](#_Toc302982766)

[5.6.7 Alternative Flow- 54](#_Toc302982767)

[5.7 Use Case: DTree Navigation/Continuation –free text input 54](#_Toc302982768)

[5.7.1 Actors 54](#_Toc302982769)

[5.7.2 Trigger 54](#_Toc302982770)

[5.7.3 Description 54](#_Toc302982771)

[5.7.4 Preconditions 54](#_Toc302982772)

[5.7.5 Postconditions 54](#_Toc302982773)

[5.7.6 Normal Flow 54](#_Toc302982774)

[5.7.7 Alternative Flow- 55](#_Toc302982775)

[5.8 Use Case: DTree Reverse Navigation –Back button/link 55](#_Toc302982776)

[5.8.1 Actors 55](#_Toc302982777)

[5.8.2 Trigger 55](#_Toc302982778)

[5.8.3 Description 55](#_Toc302982779)

[5.8.4 Preconditions 55](#_Toc302982780)

[5.8.5 Postconditions 55](#_Toc302982781)

[5.8.6 Normal Flow 55](#_Toc302982782)

[5.8.7 Alternative Flow 1- 56](#_Toc302982783)

[5.8.8 Alternative Flow 2- 56](#_Toc302982784)

[5.9 Use Case: DTree for ICS (In Conversational Survey) 56](#_Toc302982785)

[5.9.1 Actors 56](#_Toc302982786)

[5.9.2 Trigger 56](#_Toc302982787)

[5.9.3 Description 56](#_Toc302982788)

[5.9.4 Preconditions 56](#_Toc302982789)

[5.9.5 Postconditions 56](#_Toc302982790)

[5.9.6 Normal Flow 57](#_Toc302982791)

[5.9.7 Alternative Flow 1- 57](#_Toc302982792)

[5.9.8 Alternative Flow 2- 58](#_Toc302982793)

[6 Data Logging 58](#_Toc302982794)

[6.1 Use Case: Aggregate Data Load – VEngine 63](#_Toc302982795)

[6.1.1 Actors 63](#_Toc302982796)

[6.1.2 Trigger 63](#_Toc302982797)

[6.1.3 Description 63](#_Toc302982798)

[6.1.4 Preconditions 64](#_Toc302982799)

[6.1.5 Postconditions 64](#_Toc302982800)

[6.1.6 Normal Flow 64](#_Toc302982801)

[6.1.7 Alternative Flow 1- 64](#_Toc302982802)

[6.2 Use Case: Aggregate Data real time update – VEngine 64](#_Toc302982803)

[6.2.1 Actors 64](#_Toc302982804)

[6.2.2 Trigger 64](#_Toc302982805)

[6.2.3 Description 64](#_Toc302982806)

[6.2.4 Preconditions 64](#_Toc302982807)

[6.2.5 Postconditions 64](#_Toc302982808)

[6.2.6 Normal Flow 65](#_Toc302982809)

[6.2.7 Alternative Flow 1- 65](#_Toc302982810)

[6.3 Use Case: Aggregate Data calculation – V-Portal Overnight process 65](#_Toc302982811)

[6.3.1 Actors 65](#_Toc302982812)

[6.3.2 Trigger 65](#_Toc302982813)

[6.3.3 Description 65](#_Toc302982814)

[6.3.4 Preconditions 65](#_Toc302982815)

[6.3.5 Postconditions 65](#_Toc302982816)

[6.3.6 Normal Flow 65](#_Toc302982817)

[6.3.7 Alternative Flow 1- 66](#_Toc302982818)

[7 Dynamic Answers 66](#_Toc302982819)

[7.1 Use Case: Dynamic Answer Update. 66](#_Toc302982820)

[7.1.1 Actors 66](#_Toc302982821)

[7.1.2 Trigger 66](#_Toc302982822)

[7.1.3 Description 67](#_Toc302982823)

[7.1.4 Preconditions 67](#_Toc302982824)

[7.1.5 Postconditions 67](#_Toc302982825)

[7.1.6 Normal Flow 67](#_Toc302982826)

[7.1.7 Alternative Flow 1- 67](#_Toc302982827)

[7.2 Use Case: Load suppressed Dynamic Answer and reset the flag. 67](#_Toc302982828)

[7.2.1 Actors 67](#_Toc302982829)

[7.2.2 Trigger 67](#_Toc302982830)

[7.2.3 Description 67](#_Toc302982831)

[7.2.4 Preconditions 67](#_Toc302982832)

[7.2.5 Postconditions 68](#_Toc302982833)

[7.2.6 Normal Flow 68](#_Toc302982834)

[7.3 Use Case: Dynamic Answer V-Engine reload KB/re-start. 68](#_Toc302982835)

[7.3.1 Actors 68](#_Toc302982836)

[7.3.2 Trigger 68](#_Toc302982837)

[7.3.3 Description 68](#_Toc302982838)

[7.3.4 Preconditions 68](#_Toc302982839)

[7.3.5 Postconditions 68](#_Toc302982840)

[7.3.6 Normal Flow 68](#_Toc302982841)

[8 TABLE Answers 69](#_Toc302982842)

[9 NLP Process 69](#_Toc302982843)

[9.1 Recognition Structure 69](#_Toc302982844)

[9.2 NLP Process –free text 71](#_Toc302982845)

[10 Auto Complete 74](#_Toc302982846)

[10.1 Use Case: Auto Complete request. 74](#_Toc302982847)

[10.1.1 Actors 74](#_Toc302982848)

[10.1.2 Trigger 74](#_Toc302982849)

[10.1.3 Description 74](#_Toc302982850)

[10.1.4 Preconditions 74](#_Toc302982851)

[10.1.5 Postconditions 74](#_Toc302982852)

[10.1.6 Normal Flow 75](#_Toc302982853)

[10.1.7 Alternative Flow- 75](#_Toc302982854)

[11 Request Types 75](#_Toc302982855)

[11.1 Use Case: QA link request. 80](#_Toc302982856)

[11.1.1 Actors 80](#_Toc302982857)

[11.1.2 Trigger 80](#_Toc302982858)

[11.1.3 Description 80](#_Toc302982859)

[11.1.4 Preconditions 80](#_Toc302982860)

[11.1.5 Postconditions 81](#_Toc302982861)

[11.1.6 Normal Flow 81](#_Toc302982862)

[11.1.7 Alternative Flow 1- 81](#_Toc302982863)

[11.2 Use Case: Breadcrumb navigation request. 81](#_Toc302982864)

[11.2.1 Actors 81](#_Toc302982865)

[11.2.2 Trigger 81](#_Toc302982866)

[11.2.3 Description 81](#_Toc302982867)

[11.2.4 Preconditions 81](#_Toc302982868)

[11.2.5 Postconditions 81](#_Toc302982869)

[11.2.6 Normal Flow 82](#_Toc302982870)

[11.2.7 Alternative Flow 1- 82](#_Toc302982871)

[11.3 Use Case: Activity Log request. 82](#_Toc302982872)

[11.3.1 Actors 82](#_Toc302982873)

[11.3.2 Trigger 82](#_Toc302982874)

[11.3.3 Description 82](#_Toc302982875)

[11.3.4 Preconditions 82](#_Toc302982876)

[11.3.5 Postconditions 83](#_Toc302982877)

[11.3.6 Normal Flow 83](#_Toc302982878)

[11.3.7 Alternative Flow 1- 83](#_Toc302982879)

[12 Python Script Events AND FUNCTION 83](#_Toc302982880)

[13 Testing 85](#_Toc302982881)

[13.1 Alternate Testing 85](#_Toc302982882)

[13.2 Content Testing: 86](#_Toc302982883)

[13.3 Additional Test V-engine interface 86](#_Toc302982884)

[13.3.1 Parsing for syntax check 86](#_Toc302982885)

[13.3.2 VPRestart 87](#_Toc302982886)

[13.3.3 VPBack 87](#_Toc302982887)

[13.3.4 VPRestore 87](#_Toc302982888)

[13.3.5 Active Topic Interface 87](#_Toc302982889)

[13.3.6 VPExecuteScript 88](#_Toc302982890)

[13.3.7 VPAddRecognition, VPUpdateRecognition, VPAmendRecognition 88](#_Toc302982891)

[13.3.8 VPAddCondition, VPUpdateCondition, VPAmendCondition 88](#_Toc302982892)

[13.3.9 VPAddAnswer, VPUpdateAnswer, VPAmendAnswer 88](#_Toc302982893)

[13.3.10 VPAddConditionLink, VPUpdateConditionLink, VPDeleteConditionLink 88](#_Toc302982894)

[13.3.11 VPAddAnswerLink, VPUpdateAnswerLink, VPDeleteAnswerLink 89](#_Toc302982895)

[13.3.12 VPAddDTreeObject, VPUpdateDTreeObject, VPDeleteDTreeObject 89](#_Toc302982896)

[13.3.13 VPAddConnector, VPUpdateConnector, VPDeleteConnector 89](#_Toc302982897)

[13.3.14 VPAddETV, VPUpdateETV, VPDeleteETV 89](#_Toc302982898)

[13.3.15 VPAddFunction, VPUpdateFunction, VPDeleteFunction 89](#_Toc302982899)

[13.3.16 VPUpdateGlobalScripts 89](#_Toc302982900)

[13.3.17 VPUpdateSystemScripts 89](#_Toc302982901)

[13.3.18 VPExecuteScript 90](#_Toc302982902)

[13.3.19 VPUpdateLastAnswerId 90](#_Toc302982903)

[14 Variable Filling 90](#_Toc302982904)

[15 Hierarchical Topic Support 91](#_Toc302982905)

[16 Macro Encryption/Decryption 91](#_Toc302982906)

# Scope

The scope of the document is to capture the requirements related to V-Engine. The requirements specific to V-Engine plus interactions of V-portal with V-Engine will be covered as part of this document.

# Use Case: Related QUESTIONS

The following use cases define the algorithms used to determine the four different types of Related Questions (RQs). Every Answer in the Knowledgebase will have an enumerated property for each of the four RQ types which will determine how the Answer inter-acts with the RQ selection process.

The possible enumerated values for each of the four types is identical and as follows:-

1. None/Blank
2. Recommending – The associated algorithm will be used to find related questions
3. Recommendable – Answer will participate in associated algorithm to find related questions
4. Both – The associated algorithm will be used and the Answer will participate in thealgorithm to find related questions

When an Answer is chosen, it will initiate the RQ selection process for the appropriate type if the enumerated value for that type is either Recommending or Both and the project configuration switch for the RQ type is set to on.

When an RQ algorithm is selecting Answers for which the associated questions can be returned, the enumerated value for that type must be either Recommendable or Both. Also selected answers should match the current Business Area and current channel (as per the algorithm defined in section 2.2.71).

Each QA should return the following

* + answer ID
  + recognition ID
  + hit quality (precise definition dependent on source)
  + usage %
  + ICS rating

The V-Engine will apply one or multiple algorithms (depending upon the project configuration and optional request parameters) to produce one or multiple FAQ lists which will be passed on to python scripts for further processing such assorting, integrating data from the different list, layout etc.

The rules for selection of the RQ algorithm are:

1. As specified in the project configuration
2. When in the test environment, request parameters for RQ selection will override project configuration
3. If the outcome of the 1 & 2 rule above is true for the RQ selection algorithm, and if the answer’s RQ selection recommending flag is set to true, then the algorithm will be used for RQ selection. Otherwise it will not be used.

For all algorithms to find related questions (except NLP continuation) script events will be suppressed and the additional condition for the answer will not be evaluated. All non-empty additional conditionsencounteredwhen finding related questions will be defaulted to either true or false based on project configuration.

## Use Case: Related Questions – using NLP search continuation

### Actors

End User, Virtual Assistant, V-Engine

### Trigger

User enters an input (question) and clicks on submit.

### Description

This uses case describes the scenario of Related Questions based on an extended NLP search. i.e. V-engine performs a normal NLP search to find an answer to the user’s question. Once the Answer has been found, the NLP search is continued to find other Answers up to a certain condition rank (project configuration). Questions related to these answers will also be returned along with the main answer.

Project configuration will also specify max no. of RQs to be returned by V-Engine NLP search continuation algorithm to python scripts.

### Preconditions

1. Project configuration set to use NLP search continuation for Related Questions.
2. The Answer chosen for the user input has Recommending or Both flag set for NLP search continuation.

### Postconditions

Answer along with related FAQs is displayed to the user.

### Normal Flow

1. User enters a question and clicks on submit in Virtual Assistant.
2. Virtual Assistant sends the request to V-engine for NLP processing.
3. V-Engine does NLP processing to identify the answer to the user question
4. V-Engine finds a matching answer to the user question using NLP search
5. V-Engine checks the project configuration to determine which algorithm to select to find related FAQs
6. V-Engine checks if in test environment and request parameter has NLP search continuation set, it will override the project configuration parameter for NLP search continuation.
7. Project configuration is to use NLP search to determine related FAQ and answer selected to be given for the user input has NLP search continuation recommending or both flag set
8. V-Engine continues with the NLP search to find answers using all normal NLP selection criteria (e.g. Additional Conditions, Channel and Business Area checks) and running scripts where appropriate. It also checks if the selected answer has NLP search continuation recommendable flag or both set, this check to be done before evaluation of any additional condition.
9. The search is continued up to a certain condition rank (project configuration) or up to the maximum no of FAQs required (as specified in the configuration file) whichever condition is met first.
10. V-engine determines recognitions related to the answers found in step 7
11. V-Engine selects questions belonging to the recognitions as related FAQs.
12. V-engine returns main answer along with related questions
13. V-Assistant displays the answer along with related questions

### Alternative Flow 1-

From point 8 – If selected answer has question attached, V-engine selects it as FAQ question and skips point 9& 10

### Alternative Flow2 -

From point 7- V-engine continues with NLP search to find answer up to a certain rank (project configuration). If no answer is matched, V-Engine returns main answer with no related questions for this algorithm. VA displays the answer with no related questions from this algorithm.

### Hit Rating Calculation

Rank value = (20 – rank) \* 5 + (21 – subrank) / 4

Exception if the related question was found in the same section as the given answer:

Rank value = [(20 – rank) \* 5 + (21 – subrank) / 4] / [(20 – given\_rank) \* 5 + (21 – given\_subrank) / 4]

Section level = tree depth (Level(root) = 1, Level(a.b.c) = 4, etc.)

Start section level = section level on which the answer given was found

Current section level = section level of a related question found

(The exception rule for rank value will ALWAYS be used when section option is section only or no section, as in these cases we never find results in different sections)

1. Section only/No Section

Hit rating = 100 \* rank value

1. Section and parents

Section value = (current section level – 1) / start section level

Hit rating = 100 \* (section value + rank value / start section level)

1. all

Section value = (current section level ) / (start section level + 1)

For ‘all other section list’ Section value = 1 / (start section level + 1)

Hit rating = 100 \* (section value + rank value / (start section level + 1))

## Use Case: Related Questions – Semantic Auto-suggest

### Actors

End User, Virtual Assistant, V-Engine

### Trigger

User enters aninput (question) and clicks on submit.

### Description

This uses case describes the scenario of Related Questions based on a Macro index based search i.e. V-engine does NLP search to find answer to the user question. Once answer is found, V-Engine will do index based search – macro indexing to find related questions. Related Questions (RQs) will also be return along with the main answer.

### Preconditions

Project configuration to use Macro Index based search to find related questions is set to true.

The Answer chosen for the user input has recommending or both flag set for macro indexing/index based search.

The Project Configuration specifies the maximum number of Related Questions to be returned by theIndex based searched algorithm.

The V-engine has loaded macro indexing (i.e. macro list and word list with their scores) for all alternates into the memory.

### Postconditions

Answer along with Semantic AutoSuggestRelated Questions is displayed to the user.

### Normal Flow

1. User enters a question and clicks on submit in Virtual Assistant.
2. Virtual Assistant sends the request to V-engine for NLP processing.
3. V-Engine does NLP processing to identify an Answer to the user question
4. V-Engine finds a matching answer to the user question using NLP search
5. V-Engine check the project configuration to determine which algorithm to select to find related FAQs
6. V-Engine checks if in test environment and request parameter has Semantic auto-suggest set, it will override the project configuration parameter for semantic auto-suggest.
7. Project configuration is to use Index based search (using macro indexing) to determine related questions and answer selected to be given for the user input has index based search recommending or both flag set.
8. V-Engine performs anindex-based search (macro indexing) on all the alternatesof all conditionsto produce a list of conditions with their hit rating. Hit rating of a condition is the maximum of the hit rating of all its alternates.The Conditions result set is then distilled into a result set of Recognitions. The hit rating of each recognition is calculated as the maximum of the hit ratings of all its conditions.*(refer use cases in section 3 for macro indexing and index based search).*Result set of Recognition is sorted with descending order of hit rating.
9. Selected Recognitions are sorted with descending order of the hit rating.
10. V-Engine loops through the sorted recognition list and breaks when first x recognitions are selected to be returned as related FAQs where x is the maximum no of RQs as specified in the configuration file for index based search.
11. Selection is based on following criteria
    1. Answer is marked as recommendable or both for index based search
    2. Answer belongs to current business area and channel(this includes moving up the hierarchy as explained in the below flowchart)
12. V-engine returns main answer/answer to be given for the user input along with Related Questions
13. V-Assistant displays the answer along with related questions

Note: Based on project configuration, cache for answer search result for business area and channel to be maintained at recognition level.

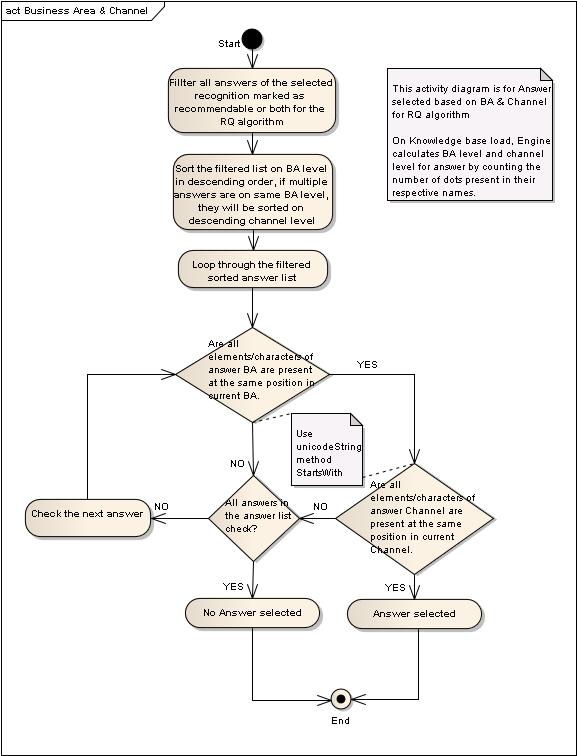
E.g. recognition id, current business area, channel, selected answer id.

### Alternative Flow -

From point 8- Index based search for related question returns zero result. It will return main answer with no FAQs. VA displays the answer back to the user but with no FAQs

#### Flowchart -

Below flowchart explains how current channel and current business area are moved up the hierarchy when answer is not in current business area and channel.



Example –

Hierarchy is maintained in dot syntax. Implementation algorithm is as follows:

Current Business area:-A.B.C.D.E

Current Channel: X.Y.Z

Selected Recognition has the following answers

|  |  |  |
| --- | --- | --- |
| Answer | Answer BA | Answer Channel |
| Answer A1 | A.B.C | X.Y |
| Answer A2 | A.B.C.D | X |
| Answer A3 | A.B.C.D | X.Y |
| Answer A4 | A.B.C.D.E | X.Y.Z.1.2 |
| Answer A5 | A.B.C.D.E.F | X.Y |
| Answer A6 | A.B | X |

Algorithm will work asfollows:

1. For the selected recognition, filter all the answers marked as recommendable or both for index based search (RQ algorithm)
2. Sort the filtered list on BA level in descending order, if multiple answers are on same BA level, they will be sorted on descending channel level.
3. Loop through the sorted list, check if
4. all element/character of answer BA is present at the same position in current BA. To do this, use Unicode string function StartsWith.
5. If above is successful, check if all element/character of answer channel is present at the same position in current channel. To do this, use Unicode string function StartsWith( in the above example Answer A3 is selected).

Note on Knowledge base load, Engine calculates answer BA level and channel by counting the number of dots present.

Above algorithm eliminates possible nesting loops for climbing up hierarchy for channel.

## Use Case: Related Questions – using FAQ category structure

### Actors

End User, Virtual Assistant, V-Engine

### Trigger

User enters aninput(question) and clicks on submit.

### Description

This uses case describes the scenario of Related Questions based on FAQ category structure. V-engine does NLP search to find answer to the user question. Once answer is found, V-Engine will find similar answers based on FAQ category. Related Questions (FAQs) will also be return along with the main answer.

### Preconditions

Project configuration to use FAQ category for related FAQs set to true.

Answer given for the user input has recommending or both flag set for FAQ category structure.

### Postconditions

Answer along with related FAQs, breadcrumb and dropdownaredisplayed to the user.

### Normal Flow

1. User enters a question and clicks on submit in Virtual Assistant.
2. Virtual Assistant sends the request to V-engine for NLP processing.
3. V-Engine does NLP processing to identify Answer to the user question
4. V-Engine finds a matching answer to the user question using NLP search
5. V-Engine checks the project configuration to determine which algorithm to select to find related questions.
6. Project configuration is to use FAQ category structure to determine related questions
7. V-Engine checks if in test environment and request parameter has FAQ category set, it will override the project configuration parameter for FAQ category set.
8. Project configuration is to use FAQ Categories to determine Related Questions and answer selected to be given for the user input has FAQ recommending or both flag set.
9. V-Engine determines the main FAQ category of the main answer, say category ‘X’ If there is no Main Category, the search for Related Questions is aborted and the currently displayed list of FAQs, the breadcrumb, and the dropdown are all unchanged.
10. V-Engine loops through all Answers and selects related FAQs.
11. The selection of related FAQs is based on the following criteria
    1. Answer has category ‘X’ as category/context
    2. Answer is marked as recommendable or both for FAQ category structure
    3. Answer belongs to current business area and channel(this includes moving up the hierarchy as explained in the flowchart section 2.2.7.1)
12. V-Engine selects questions belonging to the recognitions linked to the Answers as related FAQs.
13. V-engine returns main answer along with related FAQs and their (manually assigned) weights
14. V-Assistant displays the answer along with related FAQs

### Alternative Flow1-

From point 10– If selected answer has question attached, V-engine selects it as FAQ question and skips point 10

### Alternative Flow 2-

From point 8-V-Engine finds no answer having category ‘X’ either as main or sub category. No answer is matched, V-Engine returns main answer with no related FAQs. VA displays the answer with no related FAQs

### Algorithm to determine FAQs based on category-

Example Answers with Main Category and Category/Contexts:

|  |  |  |
| --- | --- | --- |
| **Answer** | **Main Category** | **Sub-category** |
| Answer1 | A | B, X |
| Answer2 | C | A,Y |
| Answer3 | B | A,Z |
| Answer 4 | M | X,Y,Z |

Thetablebelow shows for different input (NLP selected answer),howtherelated FAQsarefound.

|  |  |  |
| --- | --- | --- |
| **NLP selection** | **Related FAQ answer selection** | **Comments** |
| Answer1 | Answer 2 and Answer 3 | Answer 1 main category 'A' found in Answer 2&Answer 3 |
| Answer2 | no match found | Answer 2 main category 'C' not found in category of other answers. |
| Answer3 | Answer 1 | category 'B' found in answer 1 subcategory. |
| Answer4 | no match found | Answer 2 main category 'M' not found in category of other answers. |

## Related Questions – using Statistical Data

Related questions will be determined from statistical information i.e. based on the data *people who asked A also asked B.* Statistical data will be updated at the end of the user session or whenever Business Area or Channel changes for the user session.

Statistical data will be logged against Business Area and channel. Following is the structure to hold statistical information

1. Source Recognition ID
2. Source Answer ID
3. Business Area
4. Channel
5. Target Recognition ID
6. Target Answer ID
7. Count – Count will be incremented with each update.
8. Weight – with each updated, weight = weight + calculated weight for the update.
9. Last updated date

All the above parameter will be part of the key except count, weight & last updated date.

Calculation of weight is based on the following factors defined in the configuration parameter

1. Switch – on/off – switch to on/off FAQ based on statistical data
2. Bandwidth e.g. 5 - no of previous answers to consider for calculation of answer weight
3. Devaluation factor (DF) e.g. 0.75 – distance between source answer and target answer i.e.(devaluation factor)^(no of answers between source answer and target answer). *(answers with stat recommendable as NO would be considered if project flag Include non-recommendable Q/A pairs into devaluation set is True)*
4. Reverse devaluation factor (RDF) e.g. 0.6 – If target answer is an answer given to user prior to source answer.
5. Self referencing devaluation factor (SRDF) e.g. 0.1
6. Free text promotion factor (FTPF) e.g. 1.5 – If target answer is response to free text user input.
7. Clicked basic weight – basic weight to be used in calculation of answer weight.
8. Include non-recommendable Q/A pairs into devaluation set. – This flag will determine if answer with stat recommendable as NO should be considered fordevaluation set.

Answer Weight calculation - Example –

Consider following answers are given in the said sequence. All below answers belong to the same Business area and channel.

Below sequence determine the order in which answers are given to the user.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sequence | Answer ID | Request Type | Stat. Recommendable | Stat. Recommending |
| 1 | 34 | Typed | Yes | Yes |
| 2 | 65 | Category FAQ click | Yes | Yes |
| 3 | 38 | Category FAQ click | No | Yes |
| 4 | 94 | Category FAQ click | No | No |
| 5 | 105 | Typed | Yes | Yes |
| 6 | 35 | Statistical FAQ click | Yes | No |
| 7 | 234 | Typed | No | Yes |
| 8 | 54 | Statistical FAQ click | Yes | Yes |

On user session timeout or when current business area/channel is changed, statistical information of all answers in the business area and channel for the session is calculated as follows:

Project configuration 1-

1. Bandwidth = 3
2. Devaluation factor (DF) = 0.9
3. Reverse devaluation factor (RDF) = 0.6
4. Self-referencing devaluation factor (SRDF) = 0.1
5. Free Text Promotion Factor (FTPF ) =2
6. Clicked basic weight = 100
7. Include non-stat. recommendable Q/A pairs into devaluation set = true(In this case, non-recommendable Answers are included in the bandwidth selection, but are not themselves selected.However, when looking at the number of Answers between pairs for devaluation calculation, non-recommendable Answers are counted)
8. Minimum Weight – If record is not present, new record only to be added if the calculated weight is greater than Minimum weight.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **<--Target Answer Id-->** | | | | | | | |
| 34 (target answer 34 is typed, hence Free Text Promotion Factor FTPF =2) | 65 | 38 (38 has Stat. Recommendable as No | 94 (94 has Stat. Recommendable as No, hence cannot be used as target answer) | 105 (105 is typed, hence Free Text Promotion Factor FTPF =2) | 35 (35 is Statistical FAQ click, hence Self-referencing devaluation factor SRDF = 0.1) | 234  (234 has Stat. Recommendable as No, it cannot be used as target answer) | 54 (54 is Statistical FAQ click, hence Self-referencing devaluation factor SRDF = 0.1) |
| **Source Answer Id** | 34 | @ | DT=0 WT=90 (100 \*0.9) | | | | |  |  | | |  |
| 65 | DT=-0 WT=120 (100\*0.6\*2 ) | @ | | | | | DT=2 WT=145.8 (100\*0.9\*0.9\*0.9\*2) |  | | |  |
| 38 | DT=-1 WT=72 (100\*0.6\*0.6\*2 ) | DT=-0 WT=60 (100\*0.6) | @ | | | DT=1 WT=162 (100\*0.9\*0.9\*2) | DT=2 WT=7.29 (100\*0.9\*0.9\*0.9\*0.1) | | |  |
| 94 94 has Stat. Recommending as No, hence cannot be used as source answer | - | - | | | @ | - | - | | | - |
| 105 |  | DT=-2 WT=21.6 (100\*0.6\*0.6\*0.6) | | | | | @ | DT=0 WT=9 (100\*0.9\*0.1) | | | DT=2 WT=7.29 (100\*0.9\*0.9\*0.9\*0.1) |
| 35 35 has Stat. Recommending as No, hence cannot be used as source answer | - | - | | | | | - | @ | | | - |
| 234 |  |  | | | | | DT=-1 WT=72 (100\*0.6\*0.6\*2) | DT=-0 WT=6 (100\*0.6\*0.1) | @ | DT=0 WT=9 (100\*0.9\*0.1) |
| 54 |  |  | | | | | DT=-2 WT=43.2 (100\*0.6\*0.6\*0.6\*2) | DT=-1 WT=3.6 (100\*0.6\*0.6\*0.1) | | | @ |

Project Configuration 2-

1. Bandwidth = 3
2. Devaluation factor (DF) = 0.9
3. Reverse devaluation factor (RDF) = 0.6
4. Self-referencing devaluation factor (SRDF) = 0.1
5. Free Text Promotion Factor (FTPF ) =2
6. Clicked basic weight = 100
7. **Include non-stat. recommendable Q/A pairs into devaluation set = False**(In this case, non-recommendable Answers are NOT included in the bandwidth selection, and are not themselves selected.Also when looking at the number of Answers between pairs for devaluation calculation, non-recommendable Answers are NOT counted)
8. Minimum Weight – If record is not present, new record only to be added if the calculated weight is greater than Minimum weight.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **<--Target Answer Id-->** | | | | | | | |
| 34 (target answer 34 is typed, hence Free Text Promotion Factor FTPF =2) | 65 | 38 (38 has Stat. Recommendable as No | 94 (94 has Stat. Recommendable as No, hence cannot be used as target answer) | 105 (105 is typed, hence Free Text Promotion Factor FTPF =2) | 35 (35 is Statistical FAQ click, hence Self-referencing devaluation factor SRDF = 0.1) | 234  (234 has Stat. Recommendable as No, it cannot be used as target answer) | 54 (54 is Statistical FAQ click, hence Self-referencing devaluation factor SRDF = 0.1) |
| **Source Answer Id** | 34 | @ | DT=0 WT=90 (100 \*0.9) | | | | | DT=1 WT=162 (100\*0.9\*0.9\*2) | DT=2 WT=7.29 (100\*0.9\*0.9\*0.9\*0.1) | | |  |
| 65 | DT=-0 WT=120 (100\*0.6\*2 ) | @ | | | | | DT=0 WT=180 (100\*0.9\*2) | DT=1 WT=8.1 (100\*0.9\*0.9\*0.1) | | | DT=2 WT=7.29 (100\*0.9\*0.9\*0.9\*0.1) |
| 38 | DT=-1 WT=72 (100\*0.6\*0.6\*2 ) | DT=-0 WT=60 (100\*0.6) | @ | | | DT=0 WT=180 (100\*0.9\*2) | DT=1 WT=8.1 (100\*0.9\*0.9\*0.1) | | | DT=2 WT=7.29 (100\*0.9\*0.9\*0.9\*0.1) |
| 94 94 has Stat. Recommending as No, hence cannot be used as source answer | - | - | | | @ | - | - | | | - |
| 105 | DT=-1 WT=36 (100\*0.6\*0.6) | DT=-0 WT=60 (100\*0.6) | | | | | @ | DT=0 WT=9 (100\*0.9\*0.1) | | | DT=1 WT=8.1 (100\*0.9\*0.9\*0.1) |
| 35 35 has Stat. Recommending as No, hence cannot be used as source answer | - | - | | | | | - | @ | | | - |
| 234 |  | DT=-2 WT=21.6 (100\*0.6\*0.6\*0.6) | | | | | DT=-1 WT=72 (100\*0.6\*0.6\*2) | DT=-0 WT=6 (100\*0.6\*0.1) | @ | DT=0 WT=9 (100\*0.9\*0.1) |
| 54 |  | DT=-2 WT=21.6 (100\*0.6\*0.6\*0.6) | | | | | DT=-1 WT=72 (100\*0.6\*0.6\*2) | DT=-0 WT=6 (100\*0.6\*0.1) | | | @ |

### Use Case: Statistical information calculation and logging.

#### Actors

End User, Virtual Assistant, V-Engine

#### Trigger

Either one of the following

1. User enters an input and clicks on submit and there is change in Business Area or Channel.
2. User session times out.

#### Description

This use case describes the scenario of when and how the statistical information is collected and maintained.

#### Preconditions

Project configuration to use statistical data for related FAQs and project configuration defines Statistical weight calculation parameters.

#### Postconditions

Statistical data along with weight is calculated and persisted in database.

#### Normal Flow

1. User enters an input and click on submit
2. Virtual Assistant sends request to V-Engine, V-Engine does normal NLP/DTreeprocessing.
3. Business area or channel is changed either in the request or through python scripts.
4. Alternate to step 1 to 3 is Current user session times out.
5. V-Engine will go through each answer given to the user in the current session for the business area and channel, calculate associated weights as per project configuration weight calculation parameters/factors. *(Refer examples above for the weight calculation).*
6. For existing records, V-Engine will update the statistical db for these records to update weight = weight + new calculated weight, count = count +1 and last updated date = current date.
7. If new records, V-Engine will check if the calculated weight is greater than the minimum weight defined in the configuration file, if true new record is inserted.
8. V-Engine returns response to Virtual Assistant

#### Alternative Flow-

NA

### Use Case: Statistical information usage for Related FAQs.

#### Actors

End User, Virtual Assistant, V-Engine

#### Trigger

User enters aninputand clicks on submit.

#### Description

This use case describes the scenario of related FAQ based on statistical data. I.e. V-engine does NLP search to find answer to the user question. Once answer is found, V-Engine look into statistical data (DB contains aggregated table for ‘people who asked A also asked B’) and returns related questions

#### Preconditions

Project configuration to use statistical data for related FAQs.Answer given for the user input has recommending or both flag set for FAQ using statistical data. Statistical data is available in the stat DB

#### Postconditions

Answer along with related questions is displayed to the user.

#### Normal Flow

1. User enters a question and click on submit in Virtual Assistant.
2. Virtual Assistant sends the request to V-engine for NLP processing. (Apart from NLP request, Related Question click request can also trigger Statistical related question search)
3. V-Engine does NLP processing to search answer to the user question
4. V-Engine finds a matching answer to the user question using NLP search
5. V-Engine check the project configuration to determine which algorithm to select to find related FAQs
6. Check for Statistical auto-suggest request parameter if engine is in test environment in case it overrides the configuration file value
7. Project configuration is to use statistical data to determine related FAQ and answer selected to be given for the user input has FAQ statistical data recommending or both flag set.
8. V-Engine selects first n no of records from the statistical DB for the current business area, channel, source recognition id and source answer id. Selection returns Target Recognition ID, Target Answer ID, count and weight ordered in descending value of weight.
9. Check if target answer id satisfies following validation as per the loaded in memory KB
   1. Target answer Id still exists in the KB
   2. Target answer id is recommendable for statistical FAQ
   3. Target Recognition has question present.
   4. Target answer id belongs to the current business area and channel.

If any of the above validations fail, remove the record from the list.

1. Calculate total weight and % contribution for each record.
2. V-engine returns main answer along with Related Questions
3. V-Assistant displays the answer along with Related Questions

#### Alternative Flow-

NA

## Use Case: Related FAQ – All the 4 algorithm selected in configuration file

### Actors

End User, Virtual Assistant, V-Engine, Python scripts.

### Trigger

User enters aninput (question) and clicks on submit.

### Description

This use case describes the scenario where all the 4 algorithm are selected in the configuration file in order to find related FAQ

### Preconditions

Project configuration has all the 4 algorithm selected for related FAQsi.e.

1. NLP search continuation
2. Index bases search(macro indexing)
3. Using FAQ category
4. Using statistical data to determine related FAQ

Answer given for the user input has recommending or both flag set for above 4 algorithms.

### Postconditions

Answer along with related FAQs is displayed to the user.

### Normal Flow

1. User enters a question and click on submit in Virtual Assistant.
2. Virtual Assistant sends the request to V-engine for NLP processing.
3. V-Engine does NLP processing to search answer to the user question
4. V-Engine finds a matching answer to the user question using NLP search
5. Matching answer has recommending or both flag set for all 4 algorithms
6. V-Engine check the project configuration to determine which algorithm to select to find Related Questions

Check for auto-suggest request parameters if engine is in test environment. If set, they will override the configuration file values for auto suggest.

1. Project configuration specifies all 4 algorithms to use for Related Questionssearch.
2. V-Engine performs related FAQ search using each of the 4 algorithm to produce 4 different list of related FAQ
3. V-engine triggers python scripts to sorting and integrating Related Questions from 4 different lists into one.
4. V-engine returns main answer along with Related Questions
5. V-Assistant displays the answer along with Related Questions

# MacroIndexing&Index-BasedSearch

## Macro Indexing

The process of creating macro list and word list along with their scores for each alternate and storing this information against the alternate index is called macro indexing.

### Algorithm for creating the Macro index:

For each alternate -

1. Create aword list along with their respective scores
2. Alternates are scanned to create a list of unique words present.
3. Words present in the configuration file stop list (example words like ‘is’, ‘an’) are ignored and not added to the list.
4. Score of the word is calculated as weight (common constant for all words specified in the Configuration file)multiplied by weight multiplier
5. The Word Weight multiplier is calculated as –ln(occurrences of word in alternates/no of alternates)
6. The ID of the question/alternate is stored against the word
7. Creating macro list along with their respective scores
8. Questions and alternates arescanned against all the macros present in the knowledgebase. Macros with weight zero are ignored.
   1. For performance consideration, only macros having at least one word  
      present in the user input areevaluated. Macros containing word part  
      conditions (\*xxx, xxx\*) always need to be evaluated though.
9. Unique macro list is created along with their scores.
10. Score is calculated as weight present on the macro multiplied by the macro weight multiplier
11. The Macro weight multiplier is calculated as –ln(occurrences of macro in alternates/no of alternates)
12. The ID of the question/alternate is stored against the Macro

Example *How much is an apple*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Word list** | **Score** |  | **Macro List** | **Score** |
| How | 50 |  | %PRICE | 100 |
| Much | 50 |  | %FRUIT | 150 |
| Apple | 50 |  |  |  |

Following maps will be maintained

|  |  |  |
| --- | --- | --- |
| word\_score  A map containing all words in all alternates with their respective score. | key = word id &  value = word score | Score of the word id can be retrieved from the word\_score map |
| macro\_score  A map containing all macros in all alternates with their respective score. | key = macro id &  value = macro score | Score of the macro id can be retrieved from the macro\_score map |
| Word\_name  A map containing all words in all alternates with their respective name. | key = word id &  value = word name | name of the word id can be retrieved from the word\_name map |
| Macro\_name  A map containing all macros in all alternates with their respective name. | key = macro id &  value = macro name | name of the macro id can be retrieved from the macro\_name map |
| Word\_index  A map containing word id as key and value =list of alternate ids using that word. | key = word id &  value = list of question /alternate IDs | List of alternates can be retrieved from the word index map by using word id as key |
| macro\_index  A map containing macro id as key and value =list of alternate ids that evaluates the macro to true. | key = macro id &  value = list of question/ alternate IDs | List of alternates that evaluates the macro to true can be retrieved from macro index map. |
| Alternate\_word\_index | key =alternate id&  value = list of word ids | A map containing alternate id as key and value = list of words used in the alternate. |
| Alternate\_macro\_index | key =alternate id&  value = list of macro ids | A map containing alternate id as key and value = list of macro which evaluates to true for the alternate. |

Above structure will have an added advantage for real time macro index updates for new alternates/macros.

Index information will be held in memory and also persisted in indexDB (persistence is only applicable for engine in test environment)

Note: Test Engine will hold alternate text in memory as it will be used while creating indexes. For production engine, it is not required as production engine will only use indexes but never create them.

### Use Case: V-Portal publishes the KB with macro index information:

#### Actors

Administrator, Virtual Portal, V-Engine

#### Trigger

Administratorrequests in V-portal to publish KB.

#### Description

This uses case describes the scenario of where KB is published by V-portal, published KB contains macro indexing information, V-Engine loads KB along with macro indexing information.

#### Preconditions

Project configuration to use macro Indexing, Index DB is updated with macro indexing information.

#### Postconditions

KB loaded in V-Engine memory along with macro indexing information.

#### Normal Flow

1. Administratorrequests in V-portal to publish KB.
2. V-portal checks project configuration for macro indexing
3. V-portal reads Index DB to get macro indexing information
4. V-portal publishes KB which also contains macro indexing information.
5. V-portal sends request to V-Engine to reload KB.
6. V-Engine reads published KB containing macro indexing information, loads the KB into memory, it also loads macro indexing information into memory (maps).

#### Alternative Flow-

NA

### Use Case: User adds a new alternate in V-Portal

#### Actors

V-Portal user, Virtual Portal, V-Engine

#### Trigger

User requests in V-portal to add a new alternate (question itself is an alternate)

#### Description

This uses case describes the scenario of where user adds an alternate, V-Portal sends request to V-Engine to create its corresponding memory object. V-Engine will also update the macro indexing information.

#### Preconditions

Project configuration - use macro Indexing& real time macro indexing for only alternates ON.

#### Postconditions

V-Engine creates alternate in the memory, also updates the macro indexing information in memory as well in Index DB.

#### Normal Flow

1. User requests in V-portal to add an alternate.
2. V-portal sends add alternate request to V-Engine.
3. V-Engine adds the new alternate in memory object.
4. V-Engine checks project configuration to use macro indexing and real time macro indexing for alternates ON
5. V-Engine scans the alternate to generate word list and macro list.
6. Add alternate id as key and value as list of words used in the alternate into map alternate\_word\_index.
7. Similarly for macro, add into map alternate\_map\_index.
8. Loop through the generated word list and check if the word is already present in the word\_index map. If present, updated the list of alternate Ids stored against the key word id in word\_index\_map. If not present add it in word\_index map.
9. Loop through the generated macro list and check if the macro is already present in the macro\_index map. If present, updated the list of alternate Ids stored against the key macro id in macro\_index\_map. If not present add it in macro\_index map.
10. For all words present in word\_index map, calculate word\_score and update/add it in word\_score map
11. For all macros present in macro\_index map, calculate macro\_score and update/add it in macro\_score map
12. Update the above details in Index DB as well.

#### Alternative Flow 1-

From step 4) when real time macro indexing for only alternates is OFF, step 5 to 12 will be ignored. Macro indexing information for the new alternate will be created during batch mode. *Refer use case 3.1.6 – batch mode run to update Macro indexing*

#### Alternative Flow 2-

When user deletes an alternate,

1. V-engine will check alternate\_word\_index and alternate\_macro\_index map to get all words and macros used in the alternate.
2. For each word fetched above, V-engine checks if it is used by other alternate (usingword\_index map). If not, it is deleted from word\_score and word\_index. If it is being used by other alternate/s, update word\_index map to remove alternate id from the list of alternates against the word id.
3. Perform the above step for macro as well.
4. Delete from alternate\_word\_index and alternate\_macro\_index for the alternate id.
5. For all words present in word\_index map, calculate word\_score and update/add it in word\_score map
6. For all macros present in macro\_index map, calculate macro\_score and update/add it in macro\_score map
7. Update the above details in Index DB as well.

#### Alternative Flow 3-

When user amends an alternate,

1. V-engine will check alternate\_word\_index and alternate\_macro\_index map to get all old words and macros that were used in the alternate.
2. V-Engine scans the alternate to generate new word list and macro list
3. For the old words not used in the amended alternate, V-engine checks if it is used by other alternate (using word\_index map). If not, it is deleted from word\_score and word\_index. If it is being used by other alternate/s, update word\_index map to remove alternate id from the list of alternates against the word id.
4. For the new words in the amended alternate, check if it is already present in the word\_index map. If present, updated the list of alternate Ids stored against the key word id in word\_index\_map. If not present add it in word\_index map.
5. Perform the above 3 & 4 steps for macro as well.
6. Update alternate\_word\_index and alternate\_macro\_index map.
7. For all words present in word\_index map, calculate word\_score and update/add it in word\_score map
8. For all macros present in macro\_index map, calculate macro\_score and update/add it in macro\_score map
9. Update the above details in Index DB as well.

#### Alternative Flow 4-

When Project configuration - use macro Indexing for Answers is turned on, similar actions will have to be performed when a new Answer is added /amended/deleted.

Note: V-engine interface Add/Amend/Delete condition will check whether alternates of the conditions are added/deleted/amended and perform the above cases as applicable.

### Use Case: User adds/amends/deletes a macro in V-Portal

#### Actors

V-Portal user, Virtual Portal, V-Engine

#### Trigger

User requests in V-portal to add a new macro

#### Description

This uses case describes the scenario of where user adds a macro, V-Portal sends request to V-Engine to create its corresponding memory object. V-Engine will also update the macro indexing information.

#### Preconditions

Project configuration - use macro Indexing & real time macro indexing for alternates and macros ON. Also weight of the macro is non zero

#### Postconditions

V-Engine creates macro in the memory, also updates the macro indexing information in memory as well in Index DB.

#### Normal Flow

1. User requests in V-portal to add a new macro.
2. V-portal sends add macro request to V-Engine.
3. V-Engine adds the new macro in the memory object.
4. V-Engine checks project configuration to use macro indexing and real time macro indexing for alternates and macros ON.
5. V-engine checks if weight of the macro is non-zero value
6. V-Engine scans all alternates to check if the new macro has a match
7. For alternates for which new macro evaluates to true, add the macro id in the alternate\_macro\_index map
8. Update macro\_index map to add all the alternates which evaluates the macro to true.
9. Add entry in macro\_name map if not present
10. For all macros present in macro\_index map, calculate macro\_score and update/add it in macro\_score map
11. Update the above details in Index DB as well

#### Alternative Flow 1-

From step 4) when real time macro indexing for alternates and macros is OFF, step 5 to 10 will be ignored. Macro indexing information for the new macro will be created during batch mode. *Refer use case 3.1.6 – batch mode run to update Macro indexing*

#### Alternative Flow 2-

When user amends an existing macro condition text, the macro id is removed fromalternate\_macro\_index map. If weight of the macro is zero, remove entry from macro\_index, macro\_score and macro\_name maps. Otherwise follow step 5 onwards in the main flow.

#### Alternative Flow 3-

When user deletes an existing macro condition text or changes the weight from a non-zero value to zero, then the macro id is removed from alternate\_macro\_indexmap and macro-index map. Also remove entry from macro\_score map and macro\_name.

### Use Case: Restart V-Engine

#### Actors

V-Engine, V-Portal

#### Trigger

V-Engine is stopped and started again.

#### Description

This uses case describes the scenario of where V-Engine had stopped and started again. It will read load the KB along with the macro indexing information. It will also read the index DB and load macro indexing information.

#### Preconditions

Project configuration - use macro Indexing.

#### Postconditions

KB is loaded along with the macro indexing information. Macro indexing information with higher versions are read from the index DB and loaded into memory.

#### Normal Flow

1. V-Engine is not running, Request is send from V-portal which triggers V-Engine to start.
2. V-Engine reads the knowledge base and loads it in memory objects.
3. If Project configuration is to use macro indexing,
   1. It also loads the macro indexing information from the KB.
   2. V-Engine loads macro indexing information from the indexDBinto memory objects. Indexing information from the KB will be overwritten with the ones from indexDB

#### Alternative Flow-

NA

### Use Case: Batch mode run to update Macro indexing

#### Actors

V-Portal user, Virtual Portal, V-Engine

#### Trigger

V-portal sends a request to V-Engine to recreate all macro indexing information.

#### Description

This use case describes the scenario where real time macro indexing for alternates and Answers is OFF. V-Engine recreates all the macro indexing information on receiving the batch mode request from V-Portal.

#### Preconditions

Project configuration - use macro Indexing & Batch mode run to update Macro indexing information is ON.

#### Postconditions

V-Engine re-creates macro indexing information and also updates the indexDB database.

#### Normal Flow

1. V-portal request V-Engine to perform batch mode run to update macro indexing information.
2. V-Engine clears all maps, i.e. word\_name, macro\_name, word\_score macro score, word-index and macro-index, alternate\_word\_index and alternate\_macro\_map.
3. For each alternate, it creates word list and updates alternate\_word\_index&word\_index. For each word, check if entry is not present in word\_name map, if not add it.
4. For each alternate, it creates macro list and updates alternate\_macro\_index&macro\_index. For each macro, check if entry is not present in macro\_name map, if not add it.
5. For each word/macro present in word\_name/macro\_name map, V-Engine calculates the score and add entry in word\_score/macro\_score.
6. V-Engine clears previous macro information from IndexDB and adds the newly created macro indexing information.
7. V-Engine returns successful creation response to V-portal.

#### Alternative Flow-

If Project Configuration – use macro indexing for Answers is on (for engines running in Test mode to support similar Answer search for V-Portal users) then a separate set of indexes based on processed Answer text (URLs, etc removed) is also built.

V\_Portal while publishing the KB, can output plain answer text as well. Discussed with V\_Portaldev team, it is straightforward in V\_Portalto strip the answer text to remove html tags, urls, etc.

## Algorithm for Index based search (uses Macro indexing):

Usage: Used in algorithm to find

1. related questionsfor end user,
2. similar question search for v-portal user
3. similar answer search for v-portal user.
4. Apply macro indexing to the input to generate a list of macros and words. For performance optimization, only macros present in the index will be tested against the user input to generate list of macros matched to the user input.
5. Iterate through the lists of words and macros generated from the search term. For each word or Macro ID, iterate through the list of question/alternate IDs in the word-index and macro-index and store in a results map. If the alternate ID does not yet exist in the results map, add it then get the score for the source word/Macro and add that to the map against the alternate ID. If the ID already exists in the map, get the score for the word or Macro and add it to the total score stored in the map with each alternate ID.
6. Create a list containing conditions along with their hit rating. Hit rating of the condition is calculated as maximum score of all its alternate.
7. Calculate the hit rating for the associated recognition of the conditions
8. Recognition hit rating is calculated as maximum of all its conditionhit ratings.
9. Sort the list of recognition by hit rating (descending)
10. Return the topn recognitions (n as specified in the configuration file)

Example – Engine has below 2 alternate along with associated macro and word list as below

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **Alternate1** | **Macro Indexing ---------------->** | **Id** | **word** | **Score** |  | **Id** | **Macro** | **Score** |
| 1 | How much is an apple | 1 | how | 50 |  | 1 | %PRICE | 100 |
|  |  |  | 1 | much | 50 |  | 1 | %FRUIT | 150 |
|  |  |  | 1 | apple | 50 |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Id** | **Alternate1** | **Macro Indexing ---------------->** | **Id** | **word** | **Score** |  | **Id** | **Macro** | **Score** |
| 2 | How much is a rose | 2 | how | 50 |  | 2 | %PRICE | 100 |
|  |  |  | 2 | much | 50 |  | 2 | %FLOWER | 150 |
|  |  |  | 2 | rose | 50 |  |  |  |  |

asociated alternate along with macro and word list as below

Word Index

|  |  |  |
| --- | --- | --- |
| Word | Score | Alternates |
| apple | 50 | 1 |
| how | 50 | 1,2 |
| much | 50 | 1,2 |
| rose | 50 | 2 |

Macro Index

|  |  |  |
| --- | --- | --- |
| Macro | Score | Alternates |
| %FLOWER | 150 | 2 |
| %FRUIT | 150 | 1 |
| %PRICE | 100 | 1,2 |

User input*: how much is an orange*. Applying macro-indexing will generate below output.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Id** | **User input** | **Macro Indexing** --------------🡪 | **word** | **Score** |  | **Macro** | **Score** |
| NA | how much is an orange | how | 50 |  | %PRICE | 100 |
|  |  |  | much | 50 |  | %FRUIT | 150 |
|  |  |  | orange | 50 |  |  |  |

Match each item (words and macros) against the exiting alternate word-index and macro-indexto produce a map of alternate id and hit rating as follows

|  |  |  |  |
| --- | --- | --- | --- |
| Word | Score | Alternates | Result Map |
| how | 50 | 1,2 | 1 - 50  2 - 50 |
| much | 50 | 1,2 | 1 – 100  2 - 100 |
| orange | 50 | - | 1 – 100  2 - 100 |
| %PRICE | 100 | 1,2 | 1 – 200  2 - 200 |
| %FRUIT | 150 | 1 | 1 – 350  2 - 200 |

# Operators

## Immediately Followed by operator =

### Operator evaluation

Immediately followed by operator = will only work between 2 word conditions

Operator = evaluates to test if the left hand side word of theoperator is immediately followed by the right hand side word.

### Restrictions

Operator = cannot be used with macros, bracketed conditions (. . . )e.g. (a) = (b) or any other non-word-conditions. If used, this will result in syntaxerror.

There will be implied bracket in the condition where operator = used.

Example b|c=a would mean b|(c=a)

### Examples

Example 1

Condition: dog=cat

Pattern (dog cat) will be used for matching.

**'mouse dog cat horse'**will evaluate to true  
**'mouse dog fish cat horse'**will evaluate to false  
**'dog dogcat’** will evaluate to true

Example 2

Condition so=much

**I love you so much** will evaluate to True

**I love you so somuch** willevaluate to True

Example 3

Condition I=love=you+so

Implied brackets : (I = love = you) + so

Note if it is difficult for engine to determine above implied brackets, brackets can be mandatory as input in condition

**I love you so much**will evaluate to True

**I love you and your black dog so much** will evaluate to True

**I love your black dog, but does he love you so much**will evaluate to false.

**I hate him but I love you so** will evaluate to True

Example 4

Condition:no=no=no

Oh no no! will evaluate to false.

Oh no nono! will evaluate to true.

Oh no no, oh no nono! will evaluate to true.

## Loose Immediately Followed by operator >

### Operator evaluation

A>B (with A and B being any type of condition, including macros and bracketed condition structures) will behave exactly as (A+B)&!(A+\*+B).

### Restrictions

Operator > will work between any type of conditions.If the first Condition is triggered more than once before the second Condition, then the operator will return False (see example 3 below).

### Examples

Example 1

Condition: (Dogs/cats) > (eat+food)

Test string:- most cats eat lots of food

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** |  | most | cats | eat | lots | of | food |
| **Word position** |  | 1 | 2 | 3 | 4 | 5 | 6 |

(Dogs/cats) evaluates to true with start position 2,end position 2 i.e. T 2,2

(eat+food) evaluates to true with start position 3 , end position 6 i.e. T 3,6

Both left hand side and right hand side = T, compare the position to determine if > evaluates to true or false

End Position of left hand side should be = start position of right hand side -1.

End position of left end side is 2 which is equal to start position of right end side (3) -1

Hence condition (Dogs/cats) > (eat+food) evaluates to True.

Example 2

Condition:

(Dogs/cats) > (eat+food)

Test string: Most cats actually do eat food

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** |  | most | cats | actually | do | Eat | Food |
| **Word position** |  | 1 | 2 | 3 | 4 | 5 | 6 |

Left hand side (Dogs/cats) T2,2

Right hand side (eat+food) T5,6

Both LHS and RHS are true but 2 is not equal to 5 – 1, hence overall condition becomes False.

If a word appears multiple times in user input, output can evaluate to false.

Example 3

Condition:

dog>cat  
  
'mouse dog cat horse' evaluates to true  
'mouse dog fish cat horse' evaluates to false  
'dog dog cat' evaluates to false (dog = T1,1 cat = T3,3)

## Pipe operator with macros and bracketed structures.

### Operator evaluation

The pipe operator currently only works between simple word conditions. In the future it will be allowed to work with simple word conditions, or-conditions and macros.

Withor-conditions only the word list part is taken into account. The condition list is completely ignored. If the word list is empty the result is always false.

If a macro is used with the pipe, the result will always be false unless the macro is an or-condition, in which case the aforementioned rules apply.

The result of evaluating the pipe operator is equivalent to that of an OR’d list of words and Immediately Followed By Conditions. Compound words will be built for all possible combinations of one word from the left word list and one from the right word list. A threshold number can be given in the engine configuration. If the number of OR’dConditions to be generated exceeds that threshold then false will always be returned. An error is logged when the knowledgebase is parsed in this case. Chaining pipes is also allowed. Example (a/b)|(c|d)|(e/f)|g

### Restrictions

Operator | will only work between simple word conditions, or-conditions and  
macros that are simple word conditions or or-conditions.

### Examples

Example 1

pipe operator in simple condition police|car will expanded to policecar/(police=car)

Example 2

Example of the pipe operator with single or operator.

police|(car/plane) will be expanded to policecar/policeplane/(police=car)/(police=plane)

Above condition will be true for “There is a police car”

Example 3

When used with or condition, only the word list part ofthe or condition is used:-

condition (police/ambulance/(fire+fighters))|car will have (fire\_+fighters) ignored.

Condition will be condensed to

(police/ambulance)|car

Resulting in:-

policecar/ambulancecar/(police=car)/(ambulance=car)

Example 4

(cat/dog)|food

catfood/(cat=food)/dogfood/(dog=food)  
  
'cat eats food' evaluates to false  
'cat food' evaluates to true  
'dogfood' evaluates to true

Example 5

(dog/cat)|food|price

dogfoodprice/(dog=foodprice)/(dog=food=price)/(dogfood=price)/catfoodprice/(cat=foodprice)/(cat=food=price)/(catfood=price)  
  
'a dog eats food for a price' evaluates to false  
'dog food price is high' evaluates to true  
'catfood price is high' evaluates to true  
'dog foodprice is high' evaluates to true  
'catfoodprice is high' evaluates to true

A threshold number can be given in the engine configuration. If number of words to be generated exceeds the threshold, false will always be returned.

To calculate the number of resulting Conditions, establish the number of pipe operators and the number of OR’d words.

Take the number of OR’d or piped words on either side of a pipe operator and multiply them all together.

Then multiply by 2 ^ (number of non-bracketed pipe operators)

Example 1

(dog/cat)|food|price

2\*1\*1 = 2

(2^2)\*2 = 8

Example 2

(a/b)|(c|d)|(e/f)|g

2\*2\*2\*1 = 8

(2^3)\*8 = 64

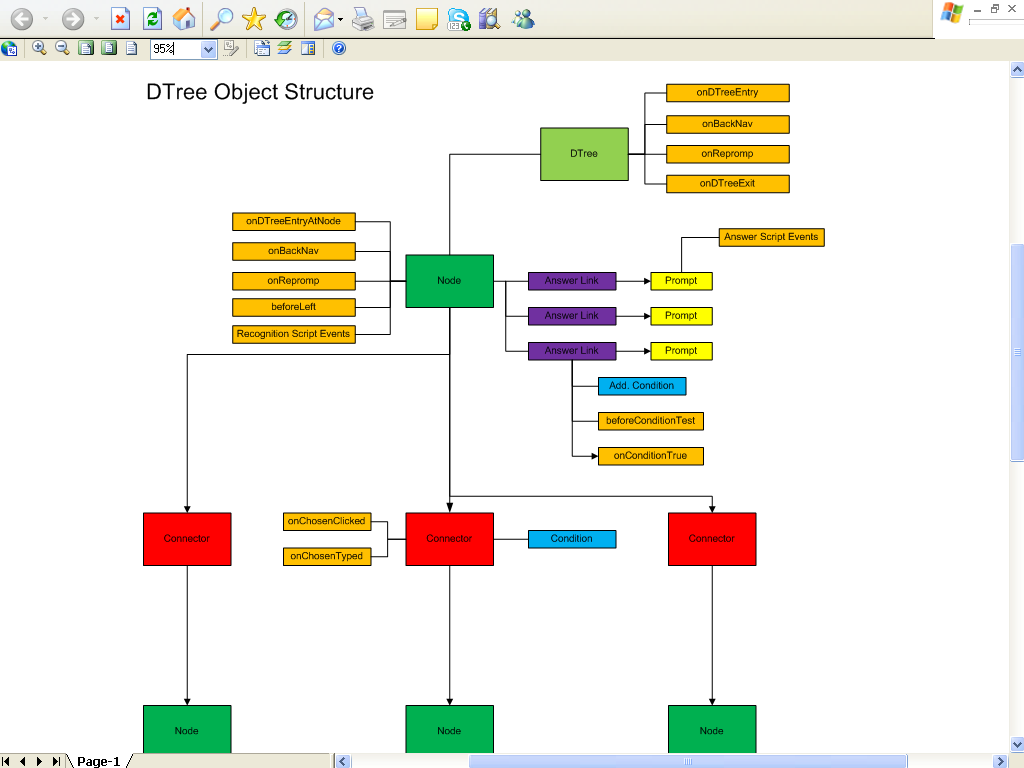
Example 3

(a/b/c)|(d/e/f)|(g/h)

3\*3\*2=18

(2^2)\*18 = 64

# Use Case: DTREE



As seen in the above D-tree structure, a D-tree contains Nodes.Anode is an object derived from the Recognition object with a few extra properties. A condition link (currently not shown in the diagram) connects a condition and a node, whilst an answer link connects an answer and a node. In DTree context, answers are called as prompts.

Nodes are connected through connectors. A node can have one or many connectors connecting to one or many nodes. Connector has direction, i.e. it connects a source node to target node. (Node 1 --> Node2).A connector also has a condition which needs to be evaluated against the user input and if true move from source node to target node. The connectors of the node are evaluated in order of the evaluation rank of the connectors.

TheDTreeobjectwill point to a node which marks start node of the DTree. The DTreeobjectwill havethe following attributes:

1. DTreeId : unique id of the DTree
2. DTree Name
3. Start node- point to the node at the top of theDTree.
4. Description – Text describing the DTree
5. Script events
   1. onDTreeEntry – On entry of the DTree
   2. onDTreeExit – On exit of the DTree
   3. onDTreeTerminate – On termination of the DTree
   4. onBackNav–When user clicks on the back button while in DTree, if the current node is not the first node in the DTree, onBackNav script will be fired.
   5. onReprompt – While in DTree, if user input doesn’t match any of the connector condition, answer is re-prompted (based on re-prompting strategy using answer rank, repeatability and MaxInvalidResponsesAllowable count at node level). On re-prompt, this script is fired.
6. Flag to disable free text input – While in DTree, user can respond either by clicking on the links provided in the answer text or by tying in the text. This flag when set, free text will be disabled, thereby forcing user to click on the link provided in the answer text
7. .ICS flag – indicating if this DTree can be used for ICS.

DTree node will have the following:

1. All recognition fields (conditions, condition links, answers(prompts), answer links etc)
2. A reference to the DTreeto which it belongs.
3. Prompt text sample
4. Flag to disable free text input –This is optional and when set will override the flag set on DTreeObjectlevel.The overridden value will only be valid to current node. Flag will specify if free text will be enabled or disabled
5. MaxInvalidResponsesAllowable count – specifies max. no of time V-engine to reprompt if user inputs text which doesn’t meet any of the connector’s condition.
6. Scripts
   1. onDTreeEntryAtNode
   2. onBackNav
   3. onReprompt -
   4. beforeLeft – This script to be fired only when a valid connector pointing to a different target node is found. V-Engine will fire this script before moving to the target node.
   5. All recognition script events
7. List of connectors. Each connector will have the following attributes
   1. condition
   2. evaluation rank – Order in which different connector of the node will be evaluated.
   3. display rank (int) – Order in which different connector link will be displayed to the end user.
   4. Scripts
      1. onConnectorChosen(script) – To be fired when the connector is used, either because the user clicks on the link provided or because its condition was triggered by the free text input.
   5. default\_click\_text (string) – display text for the link.
   6. click\_ID (int)
   7. node (DTree\_node) target node

Each user session will hold the following DTree data:

Session data

1. current DTree id
2. expected ICS DTree
3. list of active DTrees id
4. map of DTree id and DTree session data

DTree session data

1. session state of the DTree
2. path – list of DTree session state traversed- (list<DTree\_session\_state>)

Journey is the list of all the nodes traversed whereas path is the list of all nodes traversed excluding the nodes that were reversed travelled using the back button or nodes delivering multiple prompts in successiondue to re-prompting.

DTree session state

1. Current DTree node
2. Python dictionary
3. Invalid response count
4. Suggest FAQ lists

All script events of DTrees, nodes and connectors have access to the usual dictionaries (Global, System. Predefined) and additionally the DTree Dictionary of the DTree they belong to. When executed in the context of a DTree the script events of Conditions, Answer Links and Answers also get additional access to the DTreeDictionaryof the current DTree.

In the process of variable replacements on answer fields the DTreeDictionary is also taken into account, if the Answer is given in the context of a DTree.

## Use Case: DTree initialization

### Actors

End User, Virtual Assistant, V-Engine

### Trigger

User enters a question which triggers DTree

### Description

This use case describes the scenario when and how DTree gets triggered. User enters a question which will go through NLP search algorithm, NLP search algorithm will determine a condition that meets the user input. If the selected condition belongs to DTree node, DTree initialization will be triggered and DTree evaluation/navigation will begin from that node. The selected node may or may not be the start/entry node of the DTree.

### Preconditions

User gives an input for whichDTree node’s condition evaluates to true.

### Postconditions

DTree initialization is triggered, python scripts are fired. DTree is loaded into the session.

It will also be added to the list of active DTrees.

### Normal Flow

1. User enters a question and click on submit in Virtual Assistant.
2. Virtual Assistant sends the request to V-engine for NLP processing.
3. V-engine first checks in the user session if it has any currentDTree or expected ICS DTree. If no currentDTreeor expected DTree present, V-engine continues with NLP processing
4. V-Engine does NLP processing to evaluate conditionsmatching the user input
5. V-Engine selects a condition based on NLP, it checks if selected condition belongs to DTree node.
6. Selected DTree is not already present in the list of active DTrees. It is added to the list of active DTrees. Also, the DTree session is created andthe currentDTree id in the user session data is updated.
7. onDTreeEntry script of DTree object is fired.
8. V-engine fires python script onDTreeEntryAtNodefor selected node
9. V-Engine evaluates different prompts belonging to the selected node on the basis of answer link rank (normal answer search procedure of recognitions).
10. V-Engine returns the prompt to be displayed in VA to the end user.

### Alternative Flow-

From point 6

1. Selected DTree is already present in the list of active DTrees. The DTree is terminated (*refer use case 5.5 DTree Termination*). DTree session is created and updated in the user session data, current DTreeIdset. It is added in the list of active DTree.
2. onDTreeEntry script of DTree object is fired
3. V-engine fires python script onDTreeEntryAtNodefor selected node
4. V-Engine evaluates different prompts belonging to the selected node on the basis of answer link rank (normal answer search procedure of recognitions).
5. V-Engine returns the prompt to be displayed in VA to the end user.

## Use Case: DTree Entry through special request.

### Actors

End User, Virtual Assistant, V-Engine

### Trigger

User clicks a link pointing to aDTree

### Description

This use case describes the scenario where there are active DTrees in the user session, and links to these DTrees are displayed to the user in the UI. User clicks on one of the active DTree available in the VA. Special request is sent to V-Engine to set the DTree as current DTree and traverse it.

Even when DTree is not present in the active DTree list, DTree entry can be done through special request withDTree node as an optional parameter.

Following are the cases for DTree entry through special request when the DTree is not current

1. DTree present in the active DTree list. (normal flow)
2. DTree not present in the active DTree list and DTree node ID not passed as parameter. In this case DTree entered with default DTree start node (Alternate flow1)
3. DTree not present in the active DTree list and DTree node ID passed as parameter. In this case DTree entered with specifiedDTree node (Alternate flow2)

### Preconditions

User selects DTree in VA to set it as current DTree and traverse.

### Postconditions

Selected DTree is set as current DTree and traversed.

### Normal Flow

1. Engine receives special request for DTree entry with parameter as DTree Object Id
2. DTree Object Id is present in the list of active DTrees
3. If current DTree is already referring to different DTree, that DTree is exited (*refer use case 5.4 DTree Exit for steps for exiting*)
4. DTree session is updated in the user session data, current DTree Id set. DTree re-entry flag is set.
5. onDTreeEntry script of DTree object is fired
6. V-Engine evaluates different prompt belonging to the current node on the basis of answer link rank.
7. V-Engine returns the prompt to be displayed in VA to the end user.

### Alternative Flow 1-

1. Engine receives special request for DTree entry with parameter as DTreeObject Id
2. DTree Object Id is not present in the list of active DTrees.
3. If current DTree is already referring to different DTree, that DTree is exited (*refer use case 5.4 DTree Exit for steps for exiting*)
4. V-Engine initializes the DTree with default startDTree node*(refer above section 5.1forDTree Initialization)*
5. V-Engine returns the prompt to be displayed in VA to the end user.

### Alternative Flow 2-

1. Engine receives special request for DTree entry with parameter as DTree Object Id and DTreenode id
2. DTree Object Id is not present in the list of active DTrees.
3. If current DTree is already referring to different DTree, that DTree is exited (*refer use case 5.4 DTree Exit for steps for exiting*)
4. V-Engine initializes the DTree with start DTree node as passed in the parameter. *(refer above section 5.1 for DTree Initialization)*
5. V-Engine returns the prompt to be displayed in VA to the end user.

## Use Case: DTree -REPROMPT.

### Actors

End User, Virtual Assistant, V-Engine

### Trigger

User enters a input which does not match any condition of all the connectors of the current node, V-Engine will check ifMaxInvalidResponsesAllowableof the current node is not exhausted, it evaluates all the answer prompt of the current node and provide re-prompt to the user.

### Description

This use case describes the scenario where user input which does not match any condition of all the connectors of the current node, V-Engine will check if MaxInvalidResponsesAllowable of the current node is not exhausted, it re-evaluates all the answer prompt of the current node and provide re-prompt to the user

### Preconditions

V-Engine has current active DTree.

### Postconditions

A re-prompt is provided to user.

### Normal Flow

1. User enters a question and click on submit in Virtual Assistant.
2. Virtual Assistant sends the request to V-engine.
3. V-Engine before starting NLP processing, checks if there is current active DTree.
4. Current active DTree is present, therefore V-Engine first checks the user input against current DTree to find a match.
5. V-Engine checks all the connector condition of the current node against the user input.
6. None of the conditionsare evaluated to true, V-Engine checks the current node MaxInvalidResponsesAllowable count.
7. If number of invalid responses given by the user for the current node is less than or equal to MaxInvalidResponsesAllowable count, V-Engine re-evaluates all current node prompts. The previous given answer/prompt is only evaluated if its repeatability flag set to true
8. DTree level re-prompt script and after that node level re-prompt script is fired.
9. V-Engine returns the re-prompt to be displayed in VA to the end user.

### Alternative Flow-

From step 7) if number of invalid responses given by the user for the current node is greater than MaxInvalidResponsesAllowable count, DTree will be exited and NLP search will be executed for the user input. – *Refer below use case 5.4 DTree Exit*

## Use Case: DTree Exit.

### Actors

End User, Virtual Assistant, V-Engine

### Trigger

User enters a question which does not match any condition of all the connectors of the current node.

### Description

This use case describes the scenario where current DTree is exited and normal NLP search is done to process user input.

### Preconditions

V-Engine has current active DTree.

### Postconditions

Current active DTree is exited.

### Normal Flow

1. User enters a question and click on submit in Virtual Assistant.
2. Virtual Assistant sends the request to the V-engine.
3. V-Engine before starting NLP processing, checks if there is current active DTree.
4. Current active DTree is present, therefore V-Engine first checks the user input against current DTree to find a match.
5. V-Engine checks all the connector condition of the current node against the user input.
6. If none of the condition is evaluated to true, V-Engine checks the current node MaxInvalidResponsesAllowable count.
7. DTree will be exited if either of the following is true
   1. Number of invalid responses given by the user for the current node is greater than MaxInvalidResponsesAllowable count
   2. The prompt search of the node (= answer search of recognition) doesn’t return an answer.
8. The DTree script event onDTreeExit will be fired.
9. The DTreeis added in the list of active DTrees, current DTree is set to NULL
10. V-Engine starts the NLP search for the user input, on match returns the answer to be displayed in VA to the end user.

### Alternative Flow 1-

DTreewill be exited if user clicks on a QA link.

### Alternative Flow 2-

There are active DTrees in the user session, and links to these DTrees are displayed to the user in the UI. User clicks on one of the active DTree available in the VA. Special request is sent to V-Engine to set the DTree as current DTree and traverse it. If current DTree is already referring to different DTree, that DTree is exited. I.e.

1. DTree script event onDTreeExit will be fired.
2. DTree is added in the list of active DTrees, current DTree is set to NULL

## Use Case: DTreeTermination

### Actors

End User, Virtual Assistant, V-Engine

### Trigger

User enters an input which match a condition of one of the connectors of the current node, the target node of the connector is one of the terminal nodes. (A terminal node is a node with no outgoing connectors connecting to a target node). After that the user enters another question or clicks an FAQ link or the session times out.

### Description

This use case describes the scenario user input terminates current DTree.

### Preconditions

The V-Engine has a current active DTree, and the current node is connected to a terminal node through a connector.

### Postconditions

DTree is terminated. The onDTreeExit and onDTreeTerminatedeventsare fired. The current DTree in the session is set to null, and it is removed from the list of active DTrees. The DTree session and state data is cleared.

### Normal Flow

1. User enters a question and click on submit in Virtual Assistant.
2. Virtual Assistant sends the request to V-engine.
3. V-Engine before starting NLP processing, checks if there is current active DTree or expected ICS DTree.
4. Current active DTree is present, therefore V-Engine first checks the user input against current DTree to find a match.
5. V-Engine checks all the connector condition of the current node against the user input.
6. Connector condition evaluates to true and the target node of the connector is at terminal node.
7. V-Engine checks all the prompts of the target node.
8. V-Engine returns the selected prompt to be displayed in VA to the end user.
9. End user inputs another text in VA or clicks on FAQlink. The VA sends the request to V-Engine.
10. V-Engine before processing the user input, checks if current DTree is present and current node =terminal node.
11. V-Engine marks the DTree as terminated, i.e. onDTreeExit and onDTreeTerminatedeventsare fired. The current DTree in the session is set to null, it is removed from the list of active DTrees.
12. The DTree session and state data is cleared.
13. V-Engine process the user input and returns the response back to VA.

### Alternative Flow 1-

From 9) End user does not perform any action on VA, user session times out. V-Engine before expiring the user sessionwillterminate the current DTree and remove all DTrees from the list of active DTrees. Rule of thumb - whenever a DTree is removed from the active DTree list, it will be terminated. Therefore all the DTrees from the active list will be terminated as well.

### Alternative Flow 2-

From step 3

1. No current DTree or expected ICS DTree present, V-engine continues with NLP processing
2. V-Engine does NLP processing to evaluate conditions matching the user input
3. V-Engine selects a condition based on NLP, it checks if selected condition belongs to DTree node.
4. V-Engine checks if the selected DTree is already present in the list of active DTrees. If present, the DTree is terminatedi.e. onDTreeExit and onDTreeTerminatedeventsarefired. The current DTree in the session is set to null, it is removed from the list of active DTrees.
5. DTree session and state data is cleared.
6. Selected DTree is then initialized i.e.
   1. It is added to the list of active DTrees.
   2. Current DTree id in user session data is updated
   3. The onDTreeEntry script of DTree object is fired.
   4. V-engine fires python script onDTreeEntryAtNode for selected node
7. V-Engine evaluates different prompts belonging to the selected node on the basis of answer link rank (normal answer search procedure of recognitions).
8. V-Engine returns the prompt to be displayed in VA to the end user.

## Use Case: DTree Navigation/Continuation –Click Continuation

### Actors

End User, Virtual Assistant, V-Engine

### Trigger

User enters an input, DTree is initialized by V-Engine (*refer use case 4.1 DTree Initialization*)

VA displays links for the connectors of the node; user can click on any link to continue/respond.

### Description

This use case describes the navigation in DTreeusing links (not free text).

### Preconditions

DTree is initialized by V-Engine (*refer use case 5.1 DTree Initialization*) for the user input. Flag to disable free text input is set to True on the DTree object.

(Note: Flag to disable free text input set to True is not mandatory for this use case. Click continuation will be available even when disable free text input is set to False i.e. both Click continuation and free text input available to user.)

### Postconditions

One of the connector of the node is selected by the user by clicking on the link, V-Engine fires the relevant scripts and set the current node = target node of the connector selected.

### Normal Flow

1. User enters an input in VA
2. V-Engine initializes the DTree for the user input
3. V-Engine will evaluate all the prompts of the node and display the selected prompt to the user.
4. V-Engine will return the connectors of the current node to be displayed in VA as links in order of display ranks
5. User responds by clicking on one of the links displayed in VA
6. On click of the user, request is send to V-Engine with input as connector click id and request type as DTree click continuation
7. V-Engine will fire the following python scripts
   1. beforeLeft – for the current node
   2. onConnectorChosen – of the connector chosen
   3. onDTreeEntryAtNode of the target node
8. Current node is set to the target node.
9. All the prompts be evaluated and selected prompt will be displayed to user in VA
10. Also all the connectors of node will be displayed as link to the user in VA, so that user can respond by clicking on the link.

### Alternative Flow-

NA

## Use Case: DTree Navigation/Continuation –free text input

### Actors

End User, Virtual Assistant, V-Engine

### Trigger

User enters an input, DTree is initialized by V-Engine (*refer use case 4.1 DTree Initialization*)

VA displays links for the connectors of the node; also the free text input is enabled, user enters input as free text.

### Description

This use case describes the navigation in DTreefree text.

### Preconditions

DTree is initialized by V-Engine (*refer use case 4.1 DTree Initialization*) for the user input. Flag to disable free text input is set to False on the DTree object. Optionally DTree node can also specify this flag. When set, this should be set to false. (as it override the value on DTreeObject for the current node)

### Postconditions

Based on the user input (free text), all the connectors of the nodes are evaluated. Current node DTree exit event fired, selected connector script will be fired. Target node on DTree entry event will be fired and current DTree will be set to target DTree.

### Normal Flow

1. User enters an input in VA
2. V-Engine initializes the DTree for the user input
3. V-Engine will evaluate all the prompts of the node and display the selected prompt to the user.
4. V-Engine will return the connectors of the current node to be displayed in VA as links
5. As the flag to disable free text input is set to False, text input is enabled in VA
6. User will enter the free text to respond.
7. V-Engine receives the request with type as free text input
8. V-Engine evaluates all connector Conditions of the node (in order of evaluation ranks) against the user input.
9. One of the connector Conditions evaluates to true.
10. V-Engine will fire the following python scripts
    1. beforeLeft – for the current node
    2. onChosen – of the connector chosen
    3. onDTreeEntryAtNode of the target node
11. Current node is set to the target node.
12. All the prompts be evaluated and selected prompt will be displayed to user in VA
13. Also all the connectors of node will be displayed as link to the user in VA and free text enabled.

### Alternative Flow-

From step 8) none of the connectors evaluate to true, *refer use case 4.3 DTree re-prompt*

## Use Case: DTree Reverse Navigation –Back button/link

### Actors

End User, Virtual Assistant, V-Engine

### Trigger

Current Active DTree is set and user clicks on the back button

### Description

This use case describes the navigation in DTreeusinga special back button.

### Preconditions

Current Active DTree is set.

### Postconditions

Back button is clicked, onBackNav of DTree object and current DTree node fired, Path list updated to remove the current DTree.

Current DTree updated to previous DTree node in the path, invalid\_response\_count of the current DtreeNodeis reset. Current DTreeNodeis added in the Journey list.

### Normal Flow

1. User clicks on the back button in VA
2. Request of type DTree back navigation is sent to V-Engine.
3. V-Engine checks if Current active DTree is present
4. V-Engine checks the DTree session data for the path information
5. Path contains a list of DTree session state traversed (i.e. list of all nodes traversed excluding the nodes that were reversed travelled using the back button or nodes revisited due to re-prompting
6. It checks if the current node is not the first node in the path list.
7. V-Engine will fire script onBackNav of the DTree object and onBackNav on the current node.
8. V-engine will remove the current node(current DTree\_session\_state) from the path list
9. It will set the current session state = previous session state (from the path list)
10. V-Engine will update journey list to add the current node(current DTree\_session\_state).
11. Reset invalid\_response\_count of the current node, return prompt belonging to that state

### Alternative Flow 1-

From step 5) ifcurrentnode is the first node in the path, the request will be ignored.

### Alternative Flow 2-

From step 3) if current active DTree is not present, the request will be ignored.

## Use Case: DTree for ICS (In Conversational Survey)

### Actors

End User, Virtual Assistant, V-Engine

### Trigger

The answer to the user input has an ICS DTree linked. User then either enters an input that matches one of the connector of the ICS DTree or clicks on the link to select the connector of ICS DTree. ICS DTree is then initialized.

### Description

This use case describes DTreeusage for ICS.

### Preconditions

The answer to the user input has an ICS DTree linked.

### Postconditions

DTreefor ICS is initialized and user response to In- Conversational Survey (ICS)

### Normal Flow

1. User enters an input in VA
2. V-Engine checks if any current active DTree is present.
3. If no current active DTree present, V-Engine checks if it has any expected ICS DTree set.
4. If expected ICS DTree is not set, V-Engine performs the normal NLP search
5. V-Engine returns the answer/prompt to the user input.
6. Answer has a linkage to ICS DTree (DTree with ICS flag set to True)
7. V-Engine sets the expected ICS DTree and returns the ICS Dtreenode prompt along with its connectors represented as links in order of display rank.
8. Depending upon disable free text input flag, free text input will be enabled/disabled. *Refer use cases 5.6 &5.7*
9. User enters input to respond to ICS DTree node prompt. (if user clicks on the connector link, steps 10 to 12 are skipped)
10. V-Engine checks to match it against connector of current active DTree (if present).
11. If no current DTree present, it checks if expected ICS DTree is present.
12. V-Engine matches one of the connector of expected ICS DTree current node for the user.
13. V-Engine initialize the expected ICS DTree refer use case 5.1.6 –step 6 onwards
14. V-Engine sets the current node = target node of the connector.
15. V-Engine returns the prompt on the current node along with the connector links to be displayed to the user.

### Alternative Flow 1-

When a current active DTree is set:

1. User enters an input in VA
2. V-Engine checks if any current active DTree is present.
3. V-Engine evaluates all connectors of the current node of the current active DTree
4. V-Engine sets current node = target node for the selected connector.(*refer user case 5.7 DTree navigation/continuation*)
5. If target node is the terminal node and selected answer has linkage to ICS DTree, expected ICS DTree is set. Also expected ICS DTree start node prompt and all connectors are returned and displayed to the user.
6. User responds by entering input.
7. As current node is terminal node, V-Engine checks if expected ICS DTree is present and evaluates user input against all connectors of the expected ICS DTree current node.
8. V-Engine matches one of the connector of expected ICS DTree current node for the user
9. V-Engine terminates the current active DTree (*refer use case 5.5 DTree Termination steps 11 & 12*)
10. V-Engine initialize the expected ICS DTree refer use case 5.1.6 –step 6 onwards
11. V-Engine sets the current node = target node of the connector of the ICS DTree
12. V-Engine returns the prompt on the current node along with the connector links to be displayed to the user.

### Alternative Flow 2-

From step 12 of normal flow

1. V-Engine matches no connector of expected ICS DTree current node for the user input.
2. V-Engine set expected DTree to null
3. V-Engine continues with the NLP process refer section 9 NLP process

# Data Logging

Logging of Data will be at each session and transaction level. When a user logs in, a unique session id is created for that user. Each request from same user to V-Engine will be a transaction and logged under the same session id and new transaction id

1. Following new fields will be added to the session table
2. engine begin load – No of other user sessions present when user logs in
3. engine end load – No of other user sessions present when user logs out
4. browser
5. VPname–
6. kbloadtime–date and time when the KB was loaded
7. publishing ID – Publishing ID of the KB loaded into memory
8. Following new fields will be added to the transaction table
9. condition ID
10. response time – duration in micro -seconds from the request time to response time
11. expected ICS ID
12. expected ICS name
13. input type – can be either of the following
    1. free text
    2. clicked
       1. suggested FAQ
          1. from FAQ cat
          2. from statistical autosuggest
          3. from semantic autosuggest
          4. from NLP cont autosuggest
       2. free answer link
       3. free template link
       4. DTreecont
       5. DTree back navigation
14. current business area
15. current channel
16. answer type
17. Journey table
18. Session ID
19. Journey ID
20. DTree membership ID– Unique ID of the DTree object
21. DTree membership/object name– name of the DTree
22. ICS flag – flag to set if ICS DTree
23. Journey Step table
24. Journey id
25. Transaction id
26. Step no
27. Node id
28. Connector ID
29. DTree-entry flag – will be set as ‘Y’ when DTree is re-entered
30. DTreelastexit flag – will be set as ‘Y’ when DTree is exited
31. DTreelastexit reason –

* FAQ click
* maxReprompt reached

1. DTree termination flag
2. DTree termination reason

* regular termination
* session timeout
* session timeout while current

1. part of DTree path flag –Set when node is part of the path.
2. DTreereprompting flag – Will be set as ‘Y’ when the node is re-prompted.

Journey is the list of all the nodes traversed whereas path is the list of all nodes traversed excluding the nodes that were reversed travelled using the back button or nodes revisited due to re-prompting.

Example

User traverses to node 4, then clicks back button

togoback to node 3 *(Refer use case 5.8 DTree*

*reversed/back navigation)*

User then moves to node 5

So the journey Id will be as below. For node reversed travelled or re-prompted, path flag will be blank.

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Journey Id** | **Flag to specify if it is part of the path** |
| I | 1 | Y |
| Ii | 2 | Y |
| Iii | 3 | Y |
| Iv | 4 |  |
| V | 3 |  |
| Vi | 5 | Y |

|  |
| --- |
| **Path will be journey id where path flag =’Y’** |
| 1 |
| 2 |
| 3 |
| 5 |

E.g.: Current node = 4, user clicks on the back button, a new entry is logged sr.no=v with journey id=3 and path flag blank. Also as journey id 4 is no longer part of the path, hence the flag is reset to blank.

|  |  |  |
| --- | --- | --- |
| **AFTER** | | |
| **Sr. No** | **Journey Id** | **Flag to specify if it is part of the path** |
| I | 1 | Y |
| Ii | 2 | Y |
| Iii | 3 | Y |
| Iv | 4 |  |
| V | 3 |  |

|  |  |  |
| --- | --- | --- |
| **BEFORE** | | |
| **Sr. No** | **Journey Id** | **Flag to specify if it is part of the path** |
| I | 1 | Y |
| Ii | 2 | Y |
| Iii | 3 | Y |
| Iv | 4 | Y |

1. A new table (Survey table) will be created for the following fields
2. sessionID
3. transactionID
4. key – normalized or matchvariable name
5. value - normalized or matched variable value
6. soruce1 –recognition/node/condition/script
7. source2 – e.g. Match/normalized
8. answer ID–will be only set for ICS node and will refer to the Answer being surveyed
9. nodeID– node id of the current DTreenode. It will be null when not in DTreei.e. recognition.
10. SurveyID – DTreeID
11. Survey Name – DTree name

**V-portal will allow user to create normalized variable as per the project configuration.**

**For recognition &DTree node, it will be**

Key = free text

Value = combo box displaying capture user input or free text input

**For ICS Node, it will be**

Key = combo box (project configuration will provide the values.eg helpful, save\_call, etc)

Value = combo box (project configuration will provide the key dependent values.eg helpful values = YES, NO etc)

This Survey table along with the transaction table will be used to find/calculate aggregated data over a period of time as specified in the configuration.

A = Answer Count (all Answers in a transcript), from transaction table.

B=Transaction Count for allanswers with ICS ID NOT NULL

C= Transaction Count for a particular Answer with ICS ID NOT NULL

D= Count for a particular Answer where KEY is ‘HELPFUL’ (source1 = recognition/node, source2 =normalized)

E= Count for a particular Answer where KEY is ‘HELPFUL’ and VALUE is ‘YES’ (source1 = recognition/node, source2 =normalized)

F= Count for a particular Answer where KEY is ‘SAVE-CALL’ and VALUE is ‘YES’(source1 = recognition/node, source2 =normalized)

G= Count for a particular Answer where KEY is ‘SAVE-CALL’ (source1 = recognition/node, source2 =normalised)

H = Transaction Count for a particular Answer

I= Count of all transactionswith answer ID not NULL

The above aggregated count information can be used to calculate

1. Usage % = H/I
2. ICS Usage % = B/A.
3. ICS Take-up % =D/C.
4. ICS helpful %=E/D.
5. ICS Save Call %=F/G

Following is the aggregated data table structure to be created in logDB

1. Aggregated data Code – primary key
2. Aggregated data Count

The primary key is on aggregated data code.

It will be used to store overall aggregated data (not answer specific) calculated for a period as specified in the configuration file.

Answeraggregatedata table structure to be created in logDB

1. Answer ID
2. Aggregated data Code
3. Aggregated data Count

Composite primary key (answer id, aggregate data cd).

It will be used to store answer specific aggregated data calculated for a period as specified in the configuration file.

For the above tables (aggregatedata& answer aggregated data) data is aggregated for the period as specified in the project configuration file.

Likewise we also need to store aggregation count per day. Hence below 2 tables will also be created

Aggregatedatadaywisetable

1. Aggregated data Code – primary key
2. Aggregated data Count
3. date

The primary key is on aggregated data code and date

It will be used to store overall aggregated data (not answer specific) for each day

Answeraggregatedatadaywise table

1. Answer ID
2. Aggregated data Code
3. Aggregated data Count
4. date

Composite primary key (answer id, aggregate data cd, date).

It will be used to store answer specific aggregated data for each day

Lookup table to store the code /decode value of aggregated data

1. Aggregated data Code (e.g., values like A, B, C, D)
2. Aggregated data decode (e.g., values like *‘Transaction Count for all answers with ICS ID NOT NULL’)*

Note: Above table structures will be added in Engine\_Schema document.

Engine on startup or on request from V\_Portal to load aggregate data, will read data from aggregate data &answeraggregate data tables and populate objects as follows:

Two variables to hold values of A & B

Structure ‘Answer Aggregate data’ will be created to hold answer specific aggregate data

C= Transaction Count for a particular Answer with ICS ID NOT NULL

D= Count for a particular Answer where KEY is ‘HELPFUL’

E= Count for a particular Answer where KEY is ‘HELPFUL’ and VALUE is ‘YES’

F= Count for a particular Answer where KEY is ‘SAVE-CALL’ and VALUE is ‘YES’

G= Count for a particular Answer where KEY is ‘SAVE-CALL’

H = Transaction Count for a particular Answer

## Use Case: Aggregate Data Load – V-engine

### Actors

Virtual Portal, V-Engine

### Trigger

On V-Engine start or receives request from V\_Portal to reload aggregated data from table.

### Description

This use case describes how engine loadsaggregated data information.

### Preconditions

NA.

### Postconditions

Aggregated data is loaded into memory of V-engine.

### Normal Flow

1. V-engine starts or receives request from V\_Portalto reload aggregated data from logDB
2. V-engine reads the aggregated data from logDB(answer aggregated data table) and updates answer specific aggregated count information on answer object. (all answers are maintained in a map on knowledge base level)
3. V-engine reads from logDB(aggregated data table) from logDB and populates below two variables
   * A = Answer Count (all Answers in a transcript), from transaction table.
   * B=Transaction Count for all answers with ICS ID NOT NULL
4. V-engine returns successful aggregated data load message.

### Alternative Flow 1-

NA

## Use Case: Aggregate Data real time update – V-engine

### Actors

Virtual Assistant, V-Engine

### Trigger

V-engine receives request from VA to process user input.

### Description

This use case describes real time update of aggregated data by V-engine.

### Preconditions

NA.

### Postconditions

User input is processed, response generated and aggregated data is updated by V-engine.

### Normal Flow

1. V-Engine receives request from VA to process user input.
2. V-engine will process the user input (depending on the type of the request) and generate response.
3. For the selected answer, V-engine will increment the count for the following(if applicable)
   * C= Transaction Count for the Answer if ICS ID NOT NULL
   * D= Count for the Answer ifselected answer KEY is ‘HELPFUL’
   * E= Count for the Answer if selected answer KEY is ‘HELPFUL’ and VALUE is ‘YES’
   * F= Count for the Answer if selected answer KEY is ‘SAVE-CALL’ and VALUE is ‘YES’
   * G= Count for the Answer ifselected answer KEY is ‘SAVE-CALL’
   * H = Transaction Count for the Answer
4. V-engine will also increment below count information
   * A = Answer Count (all Answers in a transcript).
   * B=Transaction Count for all answers if selected answer ICS ID is not null
5. V-engine returns the response to VA.

### Alternative Flow 1-

NA

## Use Case: Aggregate Data calculation – V-Portal Overnight process

### Actors

Virtual Portal, V-Engine

### Trigger

V-portal overnight daily process triggered for aggregated data calculation.

### Description

This use case describes aggregated data calculation in overnight process of V-portal.

### Preconditions

Project configuration will specify time period over which aggregation data is to be calculated.

### Postconditions

Aggregated data is calculated and local and physical cache of V-engine & V-portal updated.

### Normal Flow

1. V-portal connects to the V-Engine logDB
2. V-portal calculates the aggregated data and answer aggregated data for the specified period of time as mentioned in the configuration file
3. V-portal also calculates aggregated data and answer aggregated data for the current day
4. V-portal will update the logDB table with aggregated count information.
5. V-portal will send request to V-Engine to load the aggregated count information from the logDB
6. V-Engine reads the logDB(aggregated data and answer aggregated data ) and updates it’s in memory with the aggregated count information.
   * V-engine will read the aggregate data table and populate in memory variables
     1. A = Answer Count (all Answers in a transcript), from transaction table.
     2. B=Transaction Count for all answers with ICS ID NOT NULL
   * V-engine will read all data from answer aggregate data table and update appropriate answer objects stored in the map.
7. V-portal also stores the aggregated count information in cache table on Knowledge Base DB.

### Alternative Flow 1-

NA

# Dynamic Answers

By default all answers will be Dynamic. Answers can be changed in V-portal and request send to V-Engine to update the answer in memory dynamically and update the answer DB.

*The business logic of update to Staging Server and Live server during Static & Dynamic Phase will be covered in V-Portal Functional Document along with their use cases.*

From the perspective of V-Engine changes for Dynamic Answer handling, request will be sent from V-portal for dynamic answer update along with the suppress flag and kb-version. Following use case explains dynamic answer update; suppress flag usage and persistence into answer DB.

(OV: we need to discuss about kb-version numbers. They are obviously somehow connected to tag names. But how exactly? This is probably mainly a v-portal issue though.) MG this will be updated once we have clarity from V-portal front.

## Use Case: Dynamic Answer Update.

### Actors

Virtual Portal, V-Engine

### Trigger

V-Portal sends request to V-Engine to update the answer.

### Description

This use case describes dynamic answer update; suppress flag usage and persistence into AnswerDB.

### Preconditions

V-portal sends request to V-Engine for dynamic answer update with suppress flag as false.

### Postconditions

V-Engine updates the answer text in memory and also persists the same in AnswerDB

### Normal Flow

1. V-Portal sends request to V-Engine to dynamic update answer with suppress flag as false.
2. V-Engine checks the suppress flag.
3. As suppress flag is false, V-Engine updates it’s in-memory answer text with the modified string
4. V-Engine also updates the answer DB with the Answer text, version number andsuppress flag information.
5. V-Engine sends successful update response back to V-portal.

### Alternative Flow 1-

If suppress flag is true, V-Engine will only update Answer DB with the Answer text, version number and suppress flag information. It will not update its in memory answer.

## Use Case: Load suppressed Dynamic Answer and reset the flag.

### Actors

Virtual Portal, V-Engine

### Trigger

V-Portal sends request to V-Engine toload all suppressed dynamic answer texts and reset the suppress flag to false.

### Description

This use case describes use case of loading all the suppressed dynamic answers.

### Preconditions

V-portal sends request to V-Engine to load all suppressed dynamic answer texts and reset the suppress flag to false

### Postconditions

V-Engine loads all the suppressed dynamic answers from answer DB into memory and reset the suppress flag

### Normal Flow

1. V-Portal sends request to V-Engine to load all suppressed dynamic answer texts and reset the suppress flag to false.
2. V-Engine reads the answer db to load all the dynamic answer texts with suppress flag as true
3. V-Engine updates the AnswerDB to update all suppress flag from false to true.
4. V-Engine gives successful response back to V-portal.

Note: On restart/reload, V-Engine will load KB. It will also load and overwrite in memory object with all the answer with suppress flag as false and with higher version than KB version from answer DB

## Use Case: Dynamic Answer V-Engine reload KB/re-start.

### Actors

Virtual Portal, V-Engine

### Trigger

V-Portal sends request to V-Engine to reload KB or V-Engine re-starts

### Description

This use case describes scenario -V-Engine reads all answers from answer DB with version equal to or higher than current KB version. It also deletes answers from AnswerDB with version lower than the KB version.

### Preconditions

V-portal sends request to V-Engine to reload KB or V-Engine re-starts

### Postconditions

V-Engine loads KB, loads answers with version equal to or higher than current KB version and suppress flag as false. It also deletes answers from AnswerDBwith versions lower than current KB version.

### Normal Flow

1. V-Portal sends request to V-Engine to re-load KB. Or V-engine re-starts.
2. V-Engine reads the KB and loads it in memory.
3. V-Engine reads all answer texts from the answer DB with suppress flag as false and version equal to or higher than current KB version.
4. V-Engine deletesanswers with lower version than KB version from AnswerDB.
5. V-Engine gives successful response back to V-portal.

# TABLE Answers

Currently table answers are stored in external file. As part of this requirement, table answer will become part of knowledge base. In V-portal, users will be able to create table answers independently, give them a unique name, and use the answer Edit page to specify whether the table answer belongs to the answer.

As part of V-Engine requirement, following are the changes

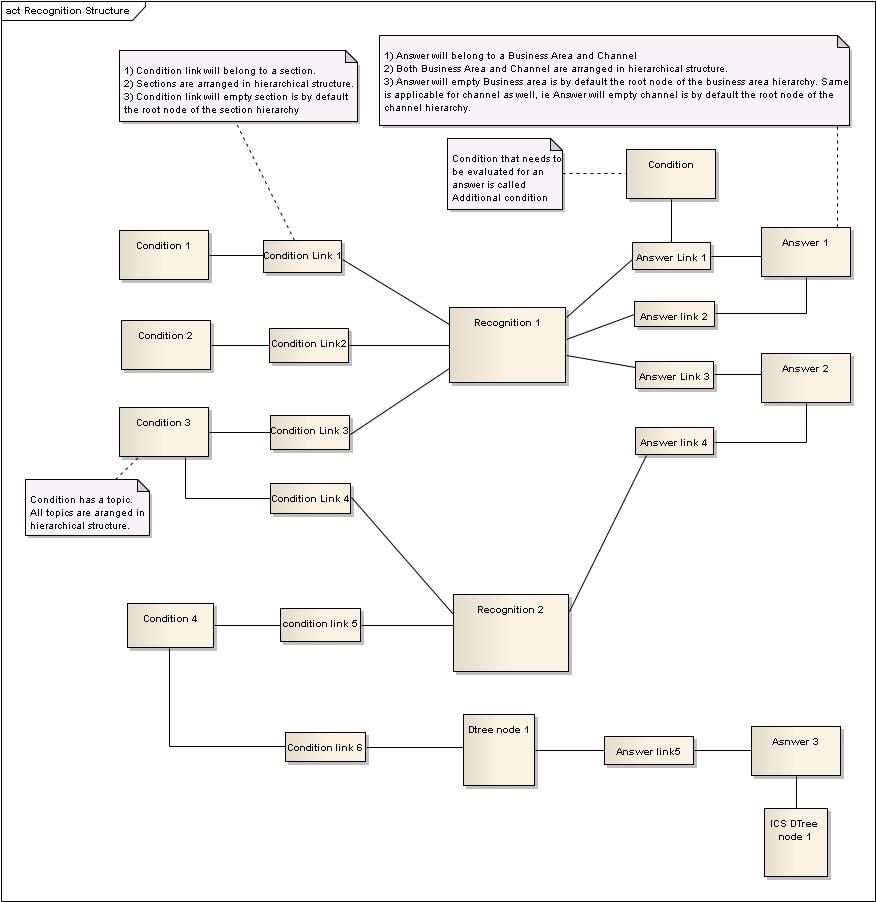
1. Ability to read table answers from the Knowledge Base (Test and Production modes)
2. Ability to accept request from V-portal to update its in memory table answers and its linkage to main answer. (Test mode only)

# NLP Process

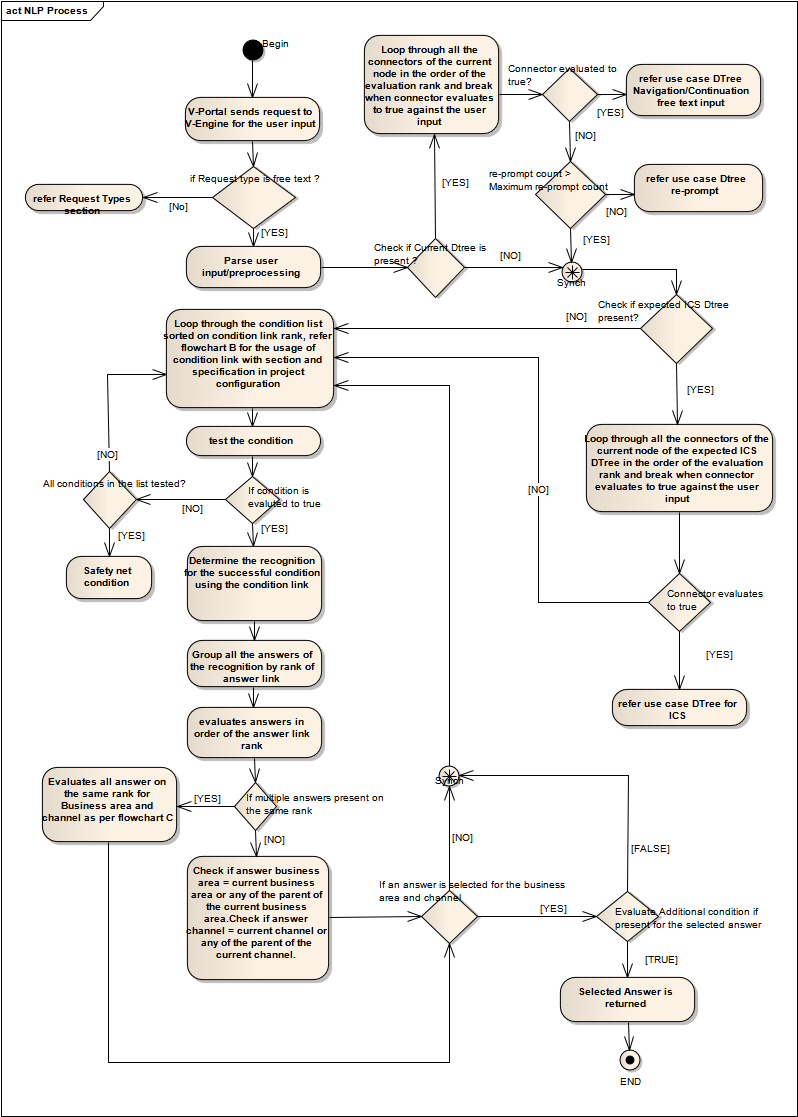
Due to the changes in KB structure as part of the requirement and addition of DTree, the current NLP process will be impact. However preprocessing & condition evaluation logic remains the same, only change to the condition evaluation is addition of the 3 new operators *(refer operator section 4)*

## Recognition Structure

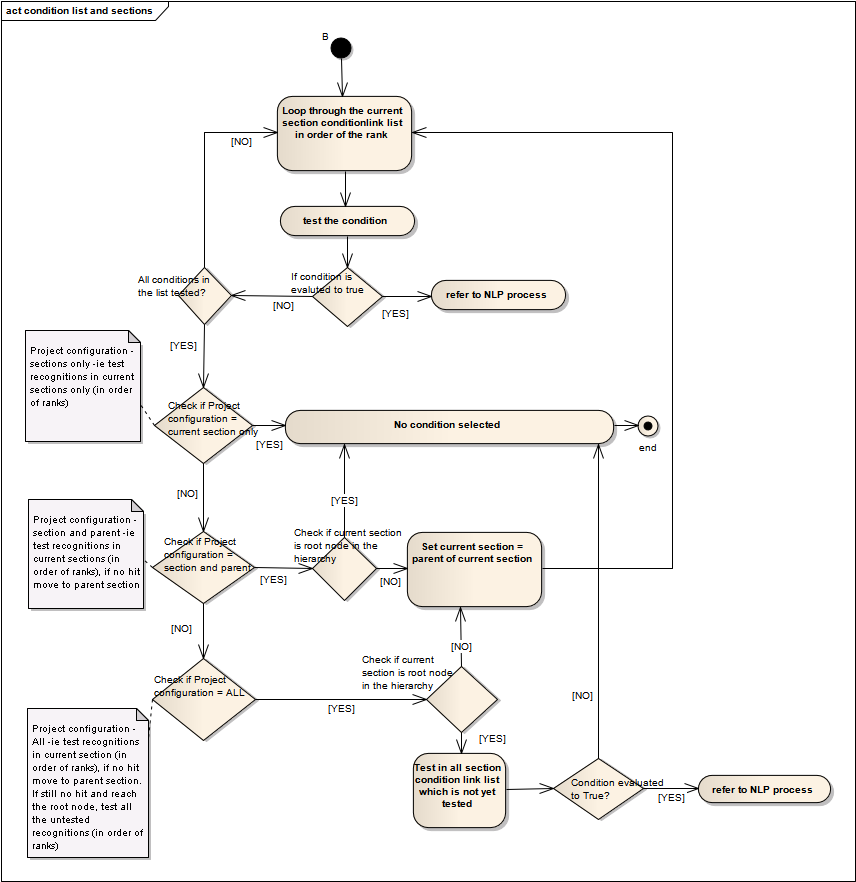
The core elements of the modified KB structures are diagrammatically represented below



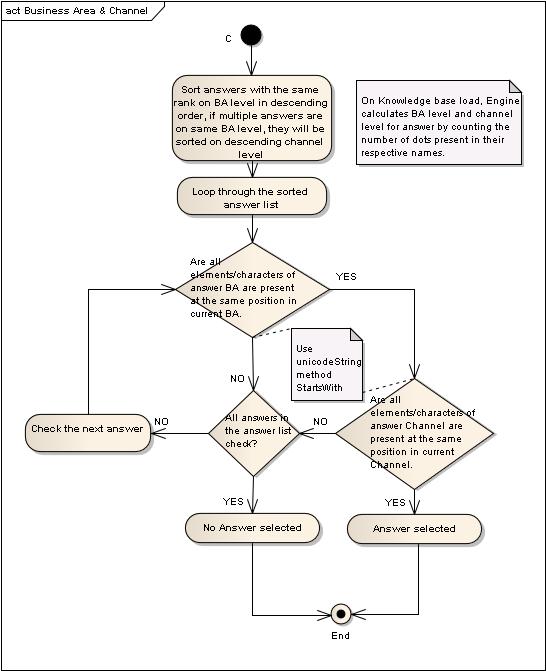
## NLP Process –free text

With the above change in the structure of recognition and its relationship with condition & answer, it association with DTree/ICS DTree, below flowchart outlines the general flow for NLP process.

B – Flowchart for condition selection process based on section and project configuration.



C – Flowchart for answer selection from a list of answers with the same rank. Selection process is based on Business area and channel.



# Auto Complete

Auto-complete will have the following 2 approaches

1. V-Assistant will send request type auto-complete with user input fragment to V-Engine, V-Engine will then search all questions which contain or start with user input fragment. For each match, following details are returned
   1. question
   2. recognition ID
   3. answer ID
   4. usage % (optional)
   5. ICS rating (optional)
2. During publish, V-portal will generate a list of each possible combination of business area and channel containing all answers belonging to the business area and channel. The Question for the answer (or the Answers parent object(s) if it has none of its own) will then be established, and this data will be passed on to the client and will be used for auto –suggest.

This use case is specific to V-portal requirement, hence will be covered in V-portal requirement document.

Below is the use case using approach 1.

## Use Case: Auto Complete request.

### Actors

Virtual Assistant, V-Engine

### Trigger

Virtual Assistant sends request to V-Engine to auto-complete user input fragment.

### Description

This use case describes Auto-complete functionality

### Preconditions

Virtual Assistant sends request to V-Engine to auto-complete user input fragment.

### Postconditions

V-Engine does case insensitive search for the user input fragment in the questions and return the following

* Question text
* Answer ID
* Recognition ID
* ICS score
* usage

### Normal Flow

1. Virtual Assistant sends a request to V-Engine to auto-complete passing user input fragment.
2. V-Engine searches all questions having at least one answer with auto-complete set as true. It does case insensitive search on questions to contain (or begin with, according to a request parameter) the user input fragment.
3. V-engine selects max n no of records as per the project configuration for auto-complete.
4. V-Engine checks if selected answers belong to the current business area and channel. Records not belonging to the current business area and channel are removed from the list.
5. V-Engine returns the following details per record to V-Assistant for auto-complete
   1. Question text
   2. Answer ID
   3. Recognition ID
   4. ICS score
   5. usage

### Alternative Flow-

NA

# Request Types

Request types other than free text are as follows:

* DTree click – It can be one of the following
  + DTree click continuation - *refer DTree user case for DTree Navigation/Continuation –Click Continuation section 5.6*
  + DTree back navigation - *refer DTree use case for DTree Reverse Navigation –Back button/link section 5.8*
  + DTree entry link – *refer DTree use case DTree entry using special request section 5.2*.
  + DTree re-entry link – Once the DTree is initialized (*refer use case DTree initialization section 5.1)* and exited *(refer use case DTree exit section 5.4)*, it can be made active/current by using this request type.
* Auto Complete – *refer Auto Complete section 10*
* QA link click
* Breadcrumb navigation – This special request type is to navigate the FAQ category. Input parameter will be FAQ category –dot syntax, possibly plus optionally cat. ID

Output:

* + Filtered below list based on selected FAQ category
    - Statistical
    - FAQ category
    - Macro index search
    - NLP cont.
  + FAQ category name
  + Possible subcategories
  + FAQ\_navigation(Combined suggest list, written by python only)
* Macro index answer search - based on the input answer text, V-Engine will perform index based search to find similar answers.
* Macro index question search - based on the input text, V-Engine will perform index based search to find similar questions
* Activity Log – This request type will be used to log data.

The input & output parameters for different request types are as follow:

**Standard input parameters in all request types:**

* Session handling items
* Business area
* Channel
* Section
* Extradata
* Ignore business area and channel (no impact on stat. Suggest)
* Suppress statistical logging

Section, Business Area and Channel when passed as input parameter i.e. not null, V-Engine should overwrite the current section, Business area and channel respectively with the values passed as input. This should be done prior to any processing and python script events.

**Standard Output in all request types:**

* Current business area
* Current channel
* Current section

Output normally means that the data is made available for replacing a template variable. For complex, structured data this is handled differently though: these are only made available to python in the predefined dictionary.

**Request types:**

* **free text input (text input)**
  + parameters:
    - user input
    - NLP on/off
    - NLP cont. on/off
    - macro index on/off (on alternates)
    - Stat. Suggest on/off (dependent on NLP on)
    - FAQ cat. Suggest on/off (dependent on NLP on)
  + Output:
    - Answer related data
* **macro index answer search**
  + parameters:
    - answer text
  + output:
    - list of similar answers
      * answer ID
      * hit rating
* **QA link (answer ID, possibly recognition ID)**
  + Parameters:
    - Recognition ID
    - Answer ID
    - Type:
      * FAQ list
        + Additional Parameter:

Subtype:

FAQ category structure

NLP continuation

stat related

macro index

* + - * template link
      * answer link
      * autocomplete text
    - Stat. Suggest on/off
    - FAQ cat. Suggest on/off
  + Output:
    - Answer related data
* **DTree click continuation**
  + parameters
    - connector ID
    - current node ID
    - current DTree ID
    - Stat. Suggest on/off
    - FAQ cat. Suggest on/off
  + Output:
    - Answer related data
* **DTree back navigation**
  + parameters
    - current node ID
    - current DTree ID
  + Output:
    - Answer related data (suggest data also stored in DTree state)
* **DTree entry link**
  + Parameters
    - DTree ID
    - optional: node ID
    - Stat. Suggest on/off
    - FAQ cat. Suggest on/off
  + Output:
    - Answer related data
* **DTree re-entry link**
  + Parameters
    - Optional: DTree ID
    - Stat. Suggest on/off
    - FAQ cat. Suggest on/off
  + Output:
    - Answer related data
* **breadcrumb navigation**
  + parameter:
    - FAQ category name (dot syntax, possibly plus optionally cat. ID)
  + Output:
    - Answer related data
* **Activity log**
  + Parameter:
    - Type (text)
    - Data (text)
  + Output
    - none
* **Auto complete**
  + Parameters:
    - User input fragment
    - Begins with/contains flag
  + Output
    - QA list
      * Question text
      * Answer ID
      * Recognition ID
      * ICS score
      * usage

Answer related data is defined as:

* Answer text
* Click continuation data if the answer is a prompt (complex)
* Answer id
* Answer link ID
* Recognition ID
* Condition link ID
* Condition ID
* Emotion
* Extra template variables
* … (everything the old engine returned)
* New output oriented or structural answer fields, also usage flags (to be reviewed when logical data model is agreed on)
* Suggested FAQ Lists (complex)
  + Statistical
  + FAQ category
  + Macro index search
  + NLP cont.
* FAQ\_navigation (Combined suggest list, written by python only)
* FAQ category name
* Possible subcategories (complex)
* Suppress free text input
* Active Topic List (only for Test Engine)
* ETVs (only for test engine)
* Vars (global & system – only for test engine)
* List of active DTrees (DTreeObject Id, DTree Node Id – only for test engine)
* Current DTreeObject Id & Current DTreeNode Id ( only for test engine)

If anything can’t be calculated within a request, the data last used in the session is repeated.

## Use Case: QA link request.

### Actors

Virtual Assistant, V-Engine

### Trigger

Virtual Assistant sends request to V-Engine for QA link click request.

### Description

This use case describes the case when user clicks on a QA link, the request is sent from VA to V-Engine which in turn returns answer related data.

### Preconditions

End user clicks on the QA link.

### Postconditions

V-Engine returns answer related data back to VA and answer is displayed in VA.

### Normal Flow

1. End user clicks on the QA link
2. Virtual Assistant sends QA link request to V-Engine passing as input parameters recognition id, answer id, link sub-type which is either RQ list (FAQ category/NLP continuation/statistical/semantic) or template link or answer link or auto complete text. Also request input will specify if Statistical suggestion, FAQ category suggestion is on/off.
3. V-Engine will return answer related data (*The contents of answer related data is already defined above)*
4. Depending on the input parameter whether Statistical suggestion is on/off, FAQ category suggestion is on/off, V-Engine will also perform related question search for these algorithms.
5. V-Engine will return 4 lists for related FAQ for 4 individual algorithms. For NLP continuation related FAQ and index based FAQ search, old list is returned. For Statistical suggestion and FAQ category suggestion based on the input parameter, new list is computed and returned or else old list will be returned.
6. V-Engine returns the result to V-Assistant

### Alternative Flow 1-

NA

## Use Case: Breadcrumb navigation request.

### Actors

Virtual Assistant, V-Engine

### Trigger

Virtual Assistant sends request to V-Engine for Breadcrumb navigation.

### Description

This use case describes Breadcrumb request functionality

### Preconditions

End user clicks on the breadcrumb link or selects one of the subcategories from the dropdown list.

### Postconditions

V-Engine returns below data back to VA

* Suggested FAQ Lists
  + Statistical (last one calculated from the session)
  + FAQ category (new one based on category selected by user)
  + Macro index search (last one calculated from the session)
  + NLP cont. (last one calculated from the session)
* Breadcrumb (With clickable elements for upward navigation)
* Possible subcategories
* FAQ\_navigation(Combined suggest list, written by python only)

### Normal Flow

1. End user clicks on the breadcrumb navigation link or selects one of the subcategories from the dropdown list
2. Virtual Assistant sends request to V-Engine for breadcrumb navigation with input parameter as FAQ category name.
3. V-Engine computes new FAQ category suggested FAQ list for the selected FAQ category (refer to Use Case: Related FAQ – using FAQ category structure) and returns it along with the 3 other lists (last ones in session history).
4. V-Engine also returns a list of subcategories( if any)
5. Virtual Assistant displays the updated FAQ list, updated breadcrumb navigation link, and updated dropdown with the new FAQ sub categories.

### Alternative Flow 1-

NA

## Use Case: Activity Log request.

### Actors

Virtual Assistant, V-Engine

### Trigger

Virtual Assistant sends request to V-Engine for activity log.

### Description

This use case describes Activity Log request. Input parameter will contain type(text) and data(text) which needs to be logged in the database.

### Preconditions

Virtual Assistant sends request to V-Engine to log activity.

### Postconditions

V-Engine logs type and data into database and returns successful response to Virtual Assistant.

### Normal Flow

1. Virtual Assistant sends request to V-Engine to log activity.
2. V-Engine logs the type(text) and data(text) into database.
3. V-Engine sends back successful update response to Virtual Assistant.

### Alternative Flow 1-

NA

# Python Script Events AND FUNCTION

Attached is the list of python events for different request types.



**New Predefined variables are as follows:**

* New items in answer related data mentioned above in the request types section, writable where possible
* DTrees (integer[])
* current\_dtree (integer)
* last\_active\_dtree (integer)
* current\_node\_ID (integer)
* invalid\_response\_count (integer)
* dtree\_termination\_reason (enumeration/string)
  + completion
  + timeout
  + restart
* dtree\_exit\_reason(enumeration/string)
  + completion
  + timeout
  + restart
  + FAQ link
  + No more reprompt
* request type(enumeration/string)
* request subtype(enumeration/string)
* Suggested FAQ Lists (possibility of complex array type and read/write to be confirmed)
  + Statistical
  + FAQ category
  + Macro index search
  + NLP cont.
* FAQ category name (writable, string)
* Possible subcategories (writable, string[])
* Auto complete data(possibility of complex array type and read/write to be confirmed)
* Activity Log Type (writable, string)
* Activity Log Data (writable, string)
* Current Business Area (, string)
* Current Channel (, string)
* Current section (, string)

New Functions required:

* SetBusinessArea (sets the current business area)
* SetChannel (sets the current channel)
* Log (logs a key value pair to the survey table)
* Function(s) for read only access to KB data (or use of predefined variables to solve this issue)

Also read access to the complete KB is required. This could be done by functions or complex predefined variables.

Read access to the data map of the object firing an event is required. This could be covered by the complete kb access though.

To give read access to all KB objects, functions such as getRecognition, getCondition, etc will be available to be called from python scripts. Function will return complex dictionary containing name value pair for simple fields and dictionaries for complex fields.

Example

GetDTreenode will return dictionary containing the following

1. DTree Object Id
2. MaxInvalidResponsesAllowable
3. FreeTextDisableFlag
4. List of Dictionary for Connectors containing
   1. EvalRank
   2. DisplayRank
   3. DefaultClickText
   4. ConnectorId
   5. DefaultConnectorText
5. Script Dictionary containing
   1. onDTreeEntryAtNode
   2. onBackNavAtNode
   3. onRepromptAtNode
   4. beforeLeft

# Testing

## Alternate Testing

Alternate testing can be executed on different levels and mode as below

**Levels**:

* alternate level: one alternate tested
* condition level: all alternates of one condition tested
* Recognition level – all alternates of all Conditions attached to a Recognition
* kb level: all alternates of all conditions in kb tested

**Modes:**

* isolated: alternate text is tested against the condition directly
* global: alternate text is handled as a user input and sent against the complete KB. For success the condition it belongs to must be the first that is triggered

Also user should be able to test a modified condition (without saving) against an input – (Local testing)

For the above alternate testing, answer selection is not required. Therefore V-engine will not to further processing after condition is evaluated to true. However interface VPUpdateLastAnwerId will be provided that can be called from V\_Portal. Therefore when in global mode, conditions that have 'must follow answer #’can verify if the last answer id matches. Similarly conditions that have topics be verified from the list of active topics. Active topic list can be updated via Topic interfaces described below section 13.3.5

Input and output parameters are as follows:

Input parameters:

1. Condition id (null in case of local testing)
2. Condition string ( null in case of alternate testing)
3. List of inputs/alternates id & text to be tested
4. Section ( null in case of local testing)
5. Node id ( to be populated for Global mode when condition belongs to DTree)
6. Mode i.e. Isolated or Global (null in case of local testing)

For global mode, when node id is passed, the context is the DTree node and not entire KB. I.e. alternates to be tested against all the connectors of that node. Connectors will be tested in order of their evaluation rank.

Output parameters:

1. List of result parameters
   1. Alternate id (null for local testing)
   2. Condition id ( null for local testing)
   3. Success = true/false

Script events will be suppressed, however, predefined variables are filled and script conditions are executed.

## Content Testing:

The test is required to simulate a live environment, wherein the content author (tester) can carry out conversations with the engine as a normal user would have. Unlike other V-Portal testing (Alternate & local), content testing is session-based.

Following request types are applicable:

* Free text
* QA Link.
* D Tree click continuation.
* D Tree back navigation.
* D Tree entry link.
* Breadcrumb navigation.

Above request type input and output parameters are already defined in section 11 Request Types

Additionally author can set the FORMVARS in V\_Portal, when set it will be passed to v-engine with each request.

## Additional Test V-engine interface

### Parsing for syntax check

V-engine interface VPParseCondition will be provided to check for syntax errors. It will receive the condition string as input(Similar to current V-Builder to Engine interface ParseCondition)

It will also be used to check syntax error for table answer condition, hence optional additional parameter implementation syntax can be passed to VPParseCondition

### VPRestart

V-Engine interface restart will be provided to clear all the session data.

### VPBack

V-Engine interface back will be provided to roll backby one conversation/transaction. It does so by removing the latest entry in thesession stack and generating the specific response for the previoustransaction/conversation.

When current DTree is set, back interface instead will trigger special DTree back functionality

### VPRestore

V-Engine interface restart will be provided to restorethe conversation. When the user closes the test dialog, the “stack” of conversations should be cleared and only the conversation one before the last conversation should be saved. On restore, last conversation is loaded and sent in the response.

### Active Topic Interface

Following interface will be provided to V\_Portal for modifying (add/update/delete) the active topic list.

#### VPAddActiveTopic

1. Input parameter: Topic Name
2. Output parameter: Success/Failure

#### VPUpdateActiveTopic

1. Input parameter: Topic Name, lifespan
2. Output parameter: Success/Failure

#### VPDeleteActiveTopic

1. Input parameter: Topic Name
2. Output parameter: Success/Failure

#### VPClearActiveTopicList

1. Input parameter: -nil
2. Output parameter: Success/Failure

### VPExecuteScript

It is required for V-engine to execute system scripts based on request from V\_Portal. Interface VPExecuteScript will be provided to execute one of the following scripts:

ONKNOWLEDGEBASELOADEDSCRIPT

ONKNOWLEDGEBASEUNLOADSCRIPT

TIMERSCRIPT

TRIGGERSCRIPT

### VPAddRecognition, VPUpdateRecognition, VPAmendRecognition

1. Input parameter: -

For add/amend - recognition object as per KB publish schema definition excluding condition link and answer link. The exclusion is made as they are separate interfaces for answerlink and condition link add/amend/delete.

For delete – recognition number.

1. Output parameter: Success/Failure

These will be reused for macro as well.

### VPAddCondition, VPUpdateCondition, VPAmendCondition

1. Input parameter: -

for add/amend - condition object as per KB publish schema definition

for delete – condition number.

1. Output parameter: Success/Failure

These will be reused for macro as well.

### VPAddAnswer, VPUpdateAnswer, VPAmendAnswer

1. Input parameter: -

for add/amend - answer object as per KB publish schema definition

for delete – answer number.

1. Output parameter: Success/Failure

### VPAddConditionLink, VPUpdateConditionLink, VPDeleteConditionLink

1. Input parameter: -

For add/amend - recognition number &conditionLink object as per KB publish schema definition

for delete – recognition number and conditionlink number.

1. Output parameter: Success/Failure

### VPAddAnswerLink, VPUpdateAnswerLink, VPDeleteAnswerLink

1. Input parameter: -

For add/amend - recognition number &answerLink object as per KB publish schema definition

For delete – recognition number and answerlink number.

1. Output parameter: Success/Failure

### VPAddDTreeObject, VPUpdateDTreeObject, VPDeleteDTreeObject

1. Input parameter: -

For add/amend –DtreeObjectas per KB publish schema definition

For delete – DtreeObject Id.

1. Output parameter: Success/Failure

### VPAddConnector, VPUpdateConnector, VPDeleteConnector

1. Input parameter: -

For add/amend – DTreeNode id and connector objectas per KB publish schema definition

For delete – DTree node id and connector id.

1. Output parameter: Success/Failure

### VPAddETV, VPUpdateETV, VPDeleteETV

1. Input parameter: -

For add/amend - ETV object as per KB publish schema definition i.e. KnowledgeBaseOptions->ExtraTemplateVariables->ETV having name and value elements

For delete – ETV name.

1. Output parameter: Success/Failure

### VPAddFunction, VPUpdateFunction, VPDeleteFunction

1. Input parameter: -

For add/amend - function object as per KB publish schema definition

For delete – Function name.

1. Output parameter: Success/Failure

### VPUpdateGlobalScripts

1. Input parameter: -GlobalScript object as per KB publish schema definition
2. Output parameter: Success/Failure

### VPUpdateSystemScripts

1. Input parameter: -SystemScript object as per KB publish schema definition.
2. Output parameter: Success/Failure

### VPExecuteScript

1. Input parameter: - Script name.
2. Output parameter: Success/Failure

### VPUpdateLastAnswerId

1. Input parameter: -Answer Id.
2. Output parameter: Success/Failure

# Variable Filling

User input variable/Matched Variable and Normalized variable will be available on condition, recognition and node level.

User input variable/Matched Variable will be populated with the user input words responsible for evaluating the condition to true. It will not be set if condition is evaluated to false or user input is clicked input. On recognition level the condition referred to here is the one that triggered the recognition. On node level the same is valid, although here two different possibilities exist: the node can be entered from outside the DTree through a condition + condition link or through a connector with condition.

Normalized variable: Normalized variable contain name-value pair and will be set in the user session if condition is evaluated to true, the recognition is triggered or a node is entered. If the value contains variable references in the ${…} syntax, the will be replaced with the current variable values(as answer field variable replacements).

Following option will be available for logging

1. Python (default option for condition/recognition): will fill the variables into the Dictionary defined below
2. Log (default option for DTree node): will log the data to the Survey table
3. Both.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Condition | Recognition | Node externally entered | Node entered through Connector |
| Requirements to be met for matched variable filling | Condition evaluated to true | Recognition triggered by one of its conditions | Node triggered by one of its conditions | Node triggered by one of the connector conditions pointing to the node |
| Value filled into match variable | user input words responsible for evaluating the condition to true | user input words responsible for evaluating the condition to true | user input words responsible for evaluating the condition to true | user input words responsible for evaluating the condition to true |
| Requirements to be met for normalised variable filling | Condition evaluated to true | Recognition triggered by one of its conditions or by directly linking to it | Node triggered by one of its conditions or by directly linking to it from outside the DTree | Node triggered by one of the connectors pointing to the node |
| Value filled into normalised variable | Value defined in KB (with variable replacement if applicable) | Value defined in KB (with variable replacement if applicable) | Value defined in KB (with variable replacement if applicable) | Value defined in KB (with variable replacement if applicable) |
| Scope to which variables are saved in option python or both | GlobalDictionary | GlobalDictionary | DTreeDictionary | DTreeDictionary |

# Hierarchical Topic Support

Information about topic’s hierarchy will be represented in dot syntax. More than one topic can be assigned to an answer.

In NLP process, condition is evaluated for the user input, selected condition topic should satisfy either of the following condition provided that previous answer given had one or more topic

1. Condition topic is equal to answer topic
2. Condition topic is with the parent structure of answer topic
   1. ex. Condition topic T1.T2, answer topic T1.T2.T3.T4, evaluation =success
   2. ex. Condition topic T1.T2.T3.T4.T5, answer topic T1.T2.T3.T4, evaluation =fail
3. Following algorithm can be used
   1. Previous answer topic will be set in session’s active topic list
   2. Check if condition topic is equal to any topic in the active topic list.
   3. If not, check if all elements/characters of condition topic is present at the same position in active topic/s (use Unicode StringStartsWith function) (note pt b & pt. c can be combined and checked with startsWith function as startsWith function also checks for equality )

# Macro Encryption/Decryption

There is a requirement on V\_Portal side to encrypt macros before persisting in database.

As entire KB file will need to be encrypted while publishing, V\_Portal will first decrypt encrypted macros and then encrypt the entire KB file while publishing.