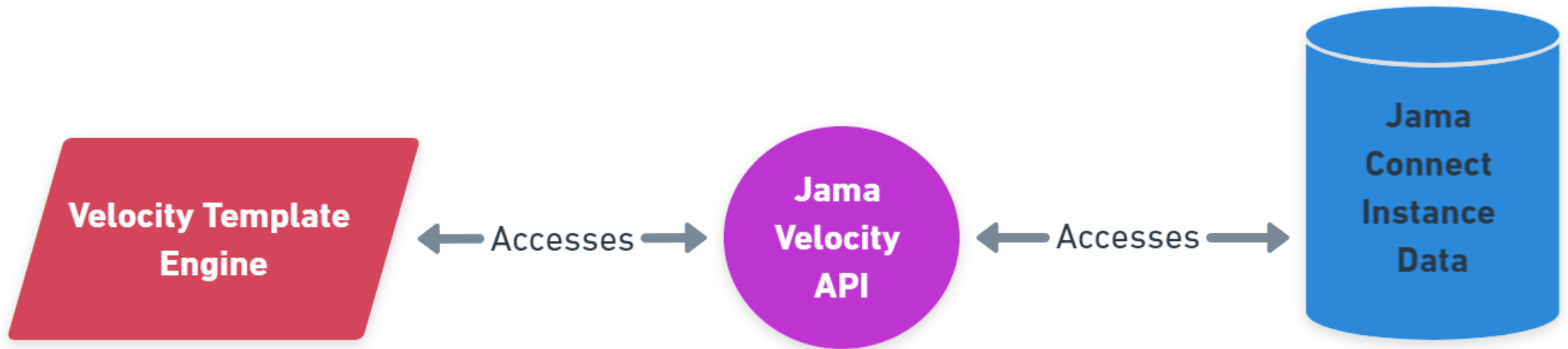


Basic Velocity Training

Part 1

What is Velocity?

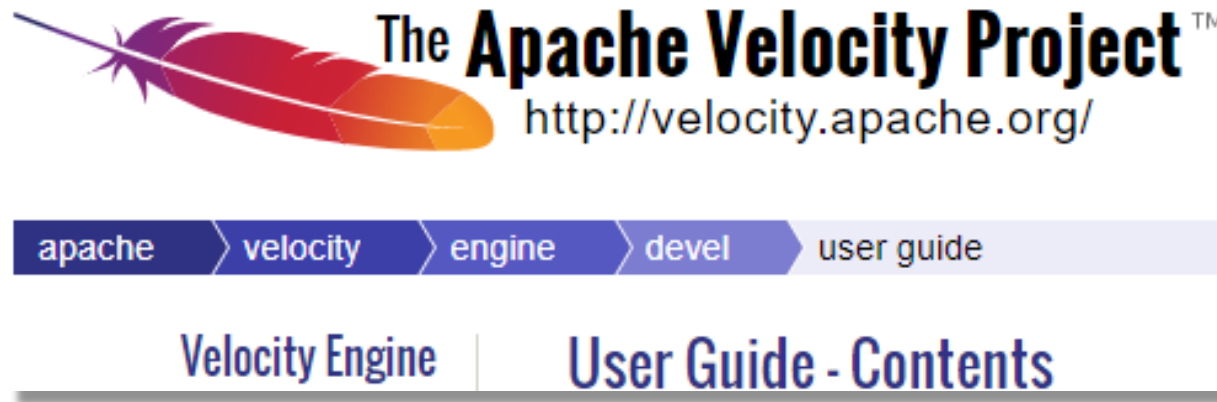
VELOCITY IS A JAVA-BASED TEMPLATE ENGINE



- Utilize the flexibility of Microsoft Word/Excel, HTML formatting, and the Velocity Engine with Java-based logic, to construct unique custom exports with a range of functionality.
- The Jama Velocity API is a Utility for accessing source code and generating reports with tools, libraries, interfaces, and built in methods defined in Java code.
- Get data from your Jama Connect instance to construct a unique export file based on your company needs

Apache Velocity Documentation

VELOCITY SYNTAX, CODE AND LOGIC



- Velocity documentation including syntax, code examples, and logic explanations, can be found on: [Apache Velocity Project](http://velocity.apache.org/).
- This will be your go to regarding the Velocity Template Language (VTL) and utilized when building custom reports.
- This training will cover report examples and code snippets to expand on this information, but the Apache documentation is where to reference the fundamentals and rules of Velocity.

Jama Velocity API

HOW TO UTILIZE THE DOCUMENTATION



Jama Velocity API Documentation

ACCESS THE JAVADOCS THAT CORRESPOND WITH YOUR JAMA CONNECT VERSION

- Utilizing Velocity, you will be able to access the [Jama Velocity API](#) to extract instance data that can be applied to your custom report definitions.
- The Velocity API documentation is organized by Jama version ranges which are dependent on if your instance is Cloud or Self-Hosted.
- Based on your Jama instance version you will utilize the corresponding API documentation option for your custom report development.
- If you are unsure of your Jama version, you will be able to find information on your Jama instance login page.



For users of Jama Connect cloud and most recent self-hosted releases.

[JavaDocs: Cloud Velocity](#)

For users of older releases of self hosted Jama Connect

[JavaDocs: 8.66 - 9.4 Velocity](#)

[JavaDocs: 8.56 - 8.66 Velocity](#)

Jama Velocity API

GETTING STARTED - JAMA SOFTWARE VELOCITY UTILITIES

- Once you select your corresponding API documentation you will be presented with the [Introduction](#) page.
- This page provides a collection of information regarding provided tools, libraries, and automatically populated global Velocity variables that can be utilized in your custom report developments.
- This page will also provide a table of each provided package available in the API which is where you will find the API methods available to you.

Packages	
Package	Description
com.jamasoftware.velocity	
com.jamasoftware.velocity.api	
com.jamasoftware.velocity.api.reporters	
com.jamasoftware.velocity.api.services	
com.jamasoftware.velocity.api.sources	This package contains the interfaces necessary to extract data out of Jama Connect for the purpose of reporting.

TOOLS

- `comparisonDateTool`
- `dateTool`
- `escapeTool`
- `loopTool`
- `mathTool`
- `numberTool`
- `sortTool`
- `valueParser`

LIBRARIES

- `VelocityReportUtil` (as `velocityReportUtil`)
- `TextUtil` (as `textUtil`)
- `DateFormatter` (as `dateFormat`)

GLOBAL VARIABLES

- **format:** The format of the report (xml, html, etc.)
- **project:** The project this report was executed against (See [Project](#))
- **baseUrl:** The base URL of your Jama Connect instance
- **report_name:** The name of the report
- **documentList:** If the report is a "Context Sensitive" report, this will be the list of documents that the user selected prior to executing the report (See: [VelocityReportDocumentDTO](#))
- **contextType:** If the report is a "Context Sensitive" report, this will be the type of list of documents that the user selected prior to executing the report (e.g. baseline, text search, etc)
- **contextId:** If the report is a "Context Sensitive" report, this will be the id of the source of the list if one is available (e.g. id of tag). For text queries, this will be the search terms used (e.g. "my search term")

Jama Velocity API

THE VELOCITY API IS ORGANIZED BY INTERFACES SPECIFIC TO JAMA DATA COLLECTIONS

- Navigating to the "Sources" package found on the [Introduction](#) page you will be presented a view of the top-level entry points into the Jama Velocity API.
- These "Source Interfaces" are categorized into primary data areas of the Jama Connect® instance and can be utilized to retrieve data that you may not have directly exported.

Interface	Description
AdminSource	Contains methods to fetch administrative information/permissions.
BaselineSource	Contains methods to fetch information regarding baselines and related information.
CommentSource	Contains methods to fetch information regarding comments.
DocumentSource	Contains methods to fetch Connect documents/items and their versions/relationships.
EventSource	
FilterSource	Contains methods to fetch information regarding filters that are saved in Connect.
ProjectSource	Contains methods to fetch information regarding the projects in Connect.
ReleaseSource	Contains methods to fetch information regarding releases.
ReviewSource	Contains methods to fetch information regarding reviews and revisions.
TestSource	Contains methods to fetch information regarding test sets, test plans, and test cycles.
UserSource	Contains methods to fetch information regarding users.
VelocityDataSource	

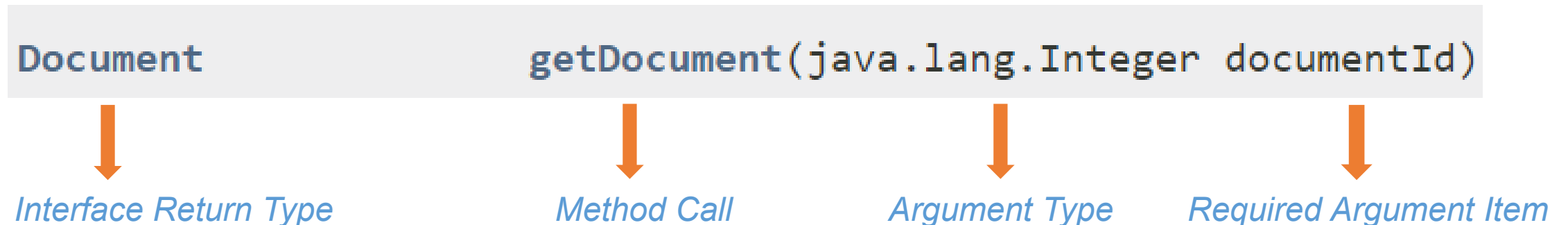
Jama Velocity API

INTERFACE METHODS AND WHAT THEY RETURN

- Each interface contains a collection of methods that can be used when you have a Velocity variable of that object type. The documentation shows the method call, arguments required (type specific), and the data type of what is returned.

Package `com.jamasoftware.velocity.api.sources`

Interface DocumentSource



Jama Velocity API Example

AN EXAMPLE OF HOW TO USE A METHOD FROM THE API WITH VELOCITY

- If we had a variable that contained a user supplied item ID and we want that item's data in the report, we can utilize the API's global interface "DocumentSource" to locate a method with a document ID as an argument, so that we can be returned a "Document" interface object.
- To call the method from our Velocity template file, we use a Java dot operator between the global interface and the method. As you can see, we accessed the global interface, called the method, supplied the argument ensuring that the variable is an integer as requested, and assigned the returned result to a new defined variable \$docItem.

```
#set($docItem = $documentSource.getDocument($mathTool.toInteger($docID)))
```

Document

getDocument(java.lang.Integer documentId)

Starting a Report From Scratch

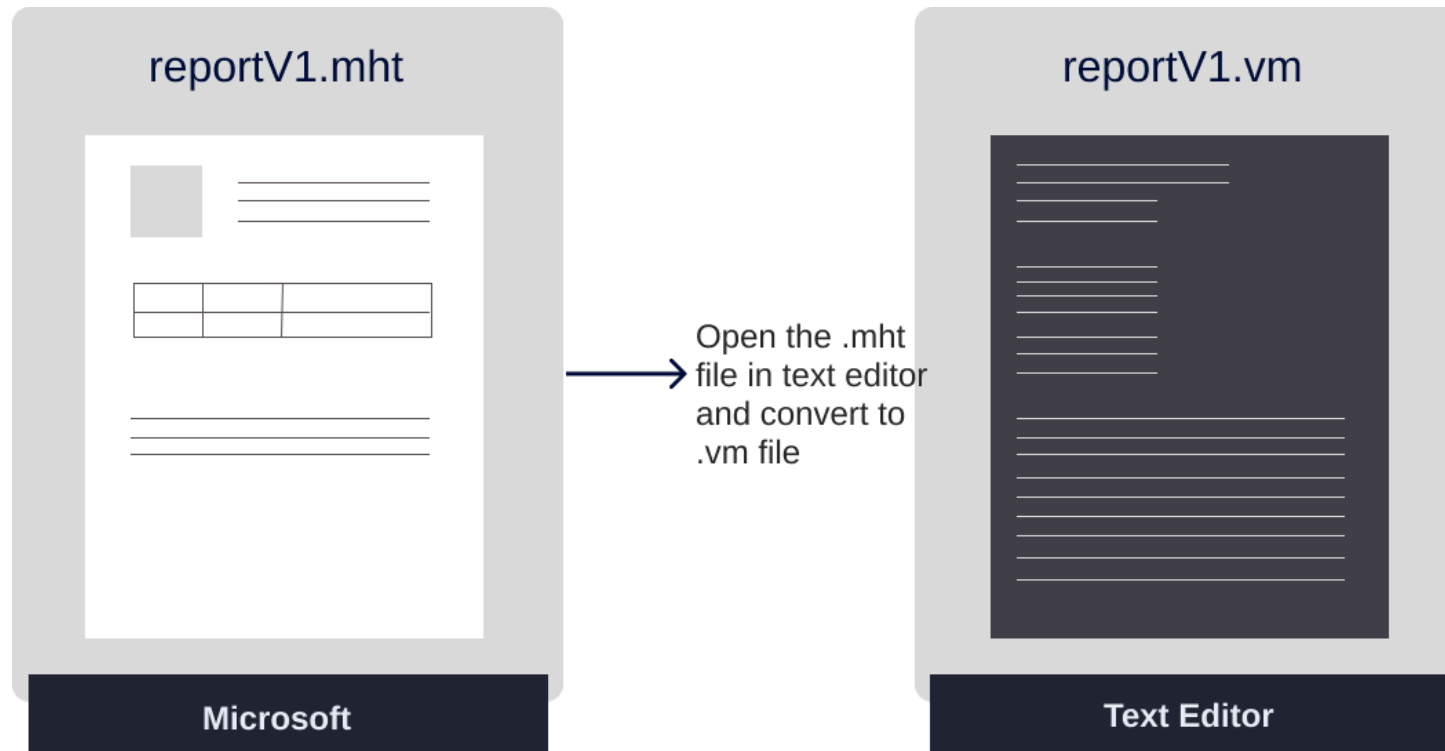
HOW TO CREATE A MOCKUP & VELOCITY FILE



Build a Mockup & Convert the File to .vm

IT IS BEST PRACTICE TO FIRST CREATE A “MOCKUP TEMPLATE”

- How would you like the report to “look” as a finished product? What does the formatting look like? How is your content structured?
- Build it out in Word or Excel, “Save As” as Single File Web Page .mht and open on your text editor.
- This .mht will show you the source HTML and CSS content that defines your created mockup which you can then extract said content to your separate .vm file instead of building it manually.



Tidy up the Velocity File

BEGIN TO ORGANIZE THE CODE OF YOUR TEMPLATE FILE

When you first open this .mht file in your text editor, there are some things to adjust in the report.

- To make the code more human readable perform the following Regex toggled find and replace actions which will realign the content and remove syntax highlight interrupting content.
- In addition, any HTML content you extract from the .mht file for your .vm report may need some minor cleaning as Microsoft can add unnecessary redundant content that adds unneeded code bulk.

Find (regex)	Replace	Change
=\n		Removes Word inserted line breaks that break up content.
=3D	=	Replace Word inserted “=3D” content which allows for easier understanding of in-line styles and better syntax highlighting.

Report Type, Installation & Parameters

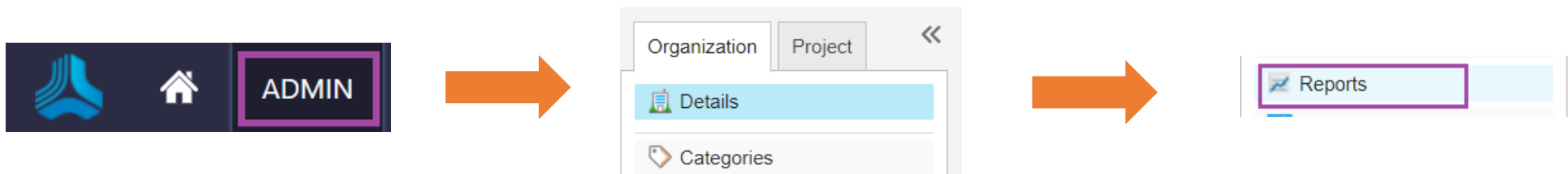
HOW TO CONFIGURE THE INSTALLATION OF YOUR REPORT



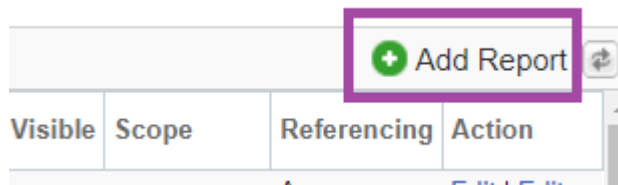
Installing A Report To Your Instance:

SELF HOSTED OR CLOUD HOSTED

1. Once a report file is ready to be installed navigate to the instance Admin > Organization > Reports.



2. Select the “Add Report” option on the top right of the screen to open the report installation menu (report installation menu detailed on next slide).



3. Once installed, to update the configuration find the row of the report installation, and, on the far right, click “Edit” to re-open the menu.



Report Installation Menu

COMPLETE REQUIRED CONFIGURATION

1. The 'Report Name' is added to the context when the report is run, and corresponds with the global constant \$report_name
2. Set the 'Report Type' as 'Velocity'
3. Select Report Formats
4. Upload the .vm file
5. Select the Organization
6. Check 'Context Sensitive' if the report will be exported from a user selected location in a project.
7. If criteria, ensure that the 'Parameter Type' and "Name" field match the variable type and name defined in your velocity report.

*Report Name:	
*Report Type:	Velocity v
*Report Formats:	☐ HTML ☐ PDF ☐ Microsoft Word ☐ Microsoft Excel
*Data Access:	☐ Limit data availability based on user permissions ☒ Allow all data to be reported on regardless of user permission
Restrict to Group:	NO GROUP RESTRICTION v
*Report File Name:	Choose File No file chosen
Description:	
*Scope:	Organization v
*Organization:	Select one... v
Visible to Users:	☒
Context Sensitive:	☐
Class name and path (optional):	
Criteria:	Select Parameter Type... v Display: Name:

Report Parameters

THERE ARE SEVERAL DATA TYPE PARAMETERS YOU CAN ADD TO A CUSTOM REPORT

- For example, a parameter can store an item ID value, which can be used to fetch data via the Velocity API, or a parameter could be a flag determining what format of the report outputs.
- They are configured as 'Criteria' in the report installation menu and are accessible to the user running the report in the report export menu UI under 'Parameters'.
- In order to use the data supplied by these parameters we reference the 'Name' field variable set in the report installation 'Criteria' menu in the report code which must be an exact match. (No typos nor trailing / leading spaces)

Visible to Users:

Context Sensitive:

Class name and path (optional):

Criteria:

Release

Location

Baseline Parameter

String Parameter

Date Parameter

Boolean

Test Plan

Review

Display: Flag Example

Name: reportBool

Add Report Criterion

Report Parameters

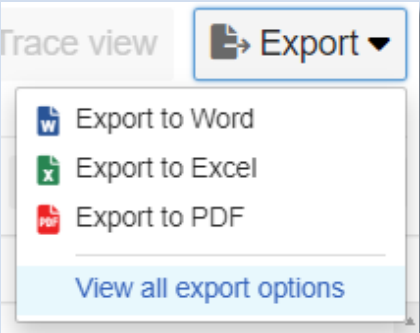
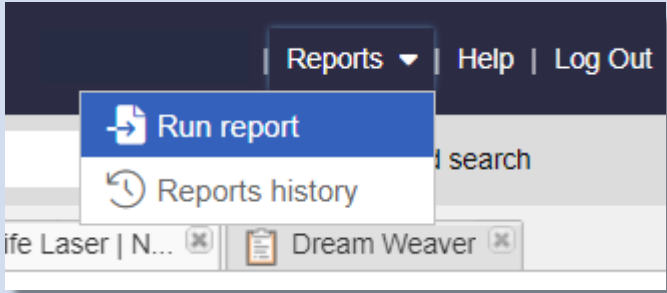
`$reportBool`

Report Context & Data Field Extraction

WHAT IS THE CONTEXT & HOW TO GET AN ITEMS DATA



Context Sensitive and In-sensitive Reports

	CONTEXT SENSITIVE	CONTEXT IN-SENSITIVE
DATA	User selected location of items in a project (Project Explorer, Baseline, Filter, Test Cycle). The items directly populate the 'Context'.	No need to select the data before exporting as data can be indirectly accessed- this can be cross project data, and at the organizational level.
VELOCITY API	The \$documentList is the global constant list that contains every item exported as VelocityReportDocumentDTO objects.	We can use a variety of Interfaces and report parameters to retrieve data for our report.
REPORT EXPORT MENU	 Export > View all export options > select the report, format, parameters > Run	 Reports > Run report > select the report, format, parameters > Run

General Context Sensitive Report Set Up

CYCLE THROUGH EACH ITEM EXPORTED

- Using a foreach loop, we cycle through the \$documentList that contains each item exported from Jama Connect. These items are [VelocityReportDocumentDTO](#) objects.
- To access the item's generic or custom field data, we utilize the VelocityReportDocumentDTO method .getDocument() which returns the item as a [ContourItem](#) Interface object type which we utilize to get the item's generic and custom field data.

```
#foreach( $vDoc in $documentList )  
  ## creates $doc, which is the document object for the  
  ## current item in the list  
  #set( $doc = $vDoc.document )  
  ## creates $docTypeKey, which is the document type for the  
  ## current item in the list  
  #set( $docTypeKey = $doc.documentType.documentTypeKey )  
  
  <p class=MsoNormal> $doc.documentKey $doc.name</p>  
  
#end
```

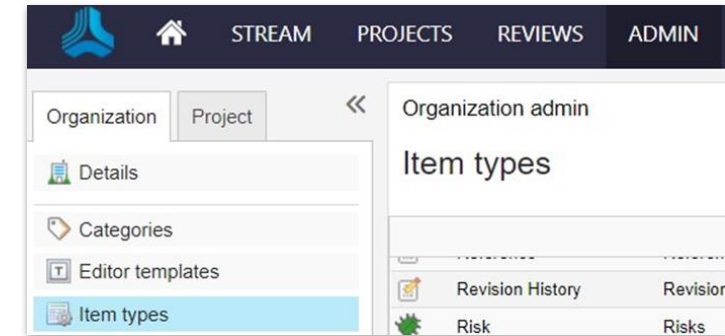


Modifier and Type	Method
ContourItem	getDocument()

Access an Item Type by “Key” or “API-ID”

WHAT ITEM TYPE/S ARE WE ACCESSING?

- Before we access the field data of any exported item, we determine what item type we have by its “Key” or “API-ID”.
- If looping through the \$documentList and the ContourItem Interface object type is set, we can use dot operators to access the [DocumentType Interface](#) methods.
 - \$doc.documentType.documentTypeKey (“Key”)
 - \$doc.documentType.id (“API-ID”)
- Once we identify the Item Type, we can proceed to properly access that Item Type’s fields.



		KEY		API-ID	
Revision History	Revision Histories	REV		158	No
Risk	Risks	RSK		151	No
Risk Control	Risk Controls	RC		152	No
Risk Evaluation	Risk Evaluations	RISK	Risk Management	145	No
Set	Sets	SET	Set type	31	Yes
Specification Document	Specification Documents	SD		153	No
Subsystem Design Output	Subsystem Design Outputs	SUBDO	Medical Device Framework	137	No
Subsystem Requirement	Subsystem Requirements	SUBSR	Medical Device Framework	138	No
SysReq	SysReqs	SYSR		149	No

Fetching Generic Fields Vs Custom Fields

PRINT AN ITEM'S FIELD DATA TO THE REPORT

- In this example, if the Item Type “Key” is equal to “SYSR” we print the exported items generic <name> field, followed by the custom field <rationale> using the globally supplied VelocityReportUtil class method .getValueForField(...)
- This method accepts your ContourItem variable, the unique field name string of your desired field found in your Admin Item Type configuration, and the globally supplied \$dateFormat variable.

```
#set( $docTypeKey == $doc.documentType.documentTypeKey )

#if( $docTypeKey == "SYSR" )

  ## Get the 'generic' field value via dot reference
  <p class=MsoNormal>${doc.name}</p>

  ## Get the 'custom' field value by utilizing $velocityReportUtil
  #set( $rationale = "" )
  #set( $rationale = $velocityReportUtil.getValueForField( $doc, "rationale", $dateFormat ) )

  <p class=MsoNormal>${rationale}</p>

#end
```

Item type fields for SysReq		
Order	Label	Unique Field Name
1	Project ID	documentKey
2	Global ID	globalId
3	Name	name
4	Description	description
6	Rationale	rationale

Empty Field Protection

AVOID EMPTY VARIABLES PRINTING IN YOUR REPORT.

- When printing item field data to the report body there may be cases where the field is empty or not set.
- In this event attempting to print your field data variable will result in the Velocity variable to appear in the report content as seen in this example of **\$notes**.
- To prevent this behavior there is a syntactical VTL solution.
 - By including a '!' after the variable syntax identifier '\$' your report variables will not appear in the report content if a field is empty / not set as seen in the example below.

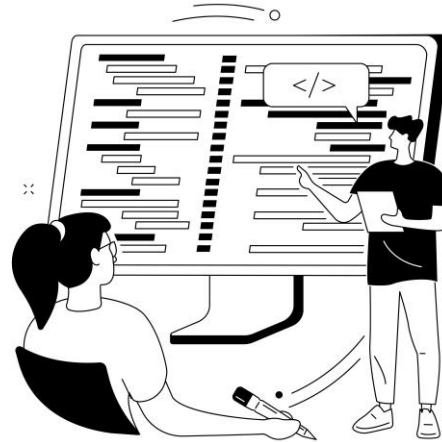
```
#set( $notes = "" )  
#set( $notes = $velocityReportUtil.getValueForField($doc, "notes", $dateFormat ) )  
<td class=x180 width=330>$!notes</td>
```

- Generic fields like .id, .documentKey, and .name, are required to have data, so you do not need to worry that they will be returned empty.

MKR ID	Notes
	\$notes
MDSD-UN-3	\$notes
MDSD-UN-2	\$notes
MDSD-UN-1	\$notes
MDSD-UN-5	\$notes
MDSD-UN-6	\$notes
MDSD-UN-3	\$notes
MDSD-UN-4	\$notes

Velocity Functional Macros

ADD REUSABLE LOGIC TO YOUR REPORT



Velocity Functional Macros

BUILD MODULAR LOGIC

- Velocity macros operate as callable functions to integrated specific logic to gather additional data, organize content, create new variables etc.
- Macros can be supplied arguments, and are called like so: `#getStatus( $doc )`
- After it is called, we can reference variables defined within the macro, like `$docStatus`.

```
## Get the doc's status value  
#getStatus($doc) ## creates $docStatus,  
$docStatus
```

```
#macro( getStatus $doc )  
  #set( $docStatus = false )  
  #foreach( $docTypeField in $doc.documentType.documentTypeFields )  
    #if( $docTypeField.documentField.name == "status" )  
      #set ( $docStatus = $doc.status.name )  
      #break  
    #end  
  #end  
#end
```

Gathering Related Items

GATHER RELATIONSHIPS FOR A PARTICULAR ITEM EITHER UP OR DOWN

The \$inDownstream argument can be true to gather downstream relationships or false to gather upstream relationships. In this example \$relDao is a variable holding the \$documentSource Data Access Object.

```
#macro( getRelationships $inDoc $inDownstream )
  #set($returnRels = [] )
  #set($rels = [] )
  #set($rels = $relDao.getRelationshipsForDocument($inDoc.id, $inDownstream))
  #foreach( $rel in $rels )
    #if( $inDownstream ) ## Get Downstream Relationships
      #if ( $rel.toDocument.active )
        #set( $success = $returnRels.add( $rel ) )
      #end
    #else ## Get Upstream Relationships
      #if ( $rel.fromDocument.active )
        #set( $success = $returnRels.add( $rel ) )
      #end
    #end
  #end
#end
```

Gathering Related Items

TO PROPERLY USE THE #GETRELATIONSHIPS MACRO YOU CAN USE THE FOLLOWING STEPS:

- Call the macro *#getRelationships* supplying the [ContourItem](#) you want to retrieve the related items for and a boolean argument that specifies to fetch either upstream (false) or downstream (true).
- This macro returns a list of [DocumentDocument](#) relationship objects, *\$returnRels*.
- For each relationship object, you will convert the [DocumentDocument](#) object into a [ContourItem](#) object by applying *.fromDocument* (upstream) or *.toDocument* (downstream) to create \$upDoc or \$dnDoc.
- Using these defined variables, you can access the related items data fields.

Upstream call

```
#getRelationships($doc, false) |
#set ( $upRels = $returnRels )
#foreach ( $upRel in $upRels )
  #set($upDoc = $upRel.fromDocument)
  $upDoc.documentKey $upDoc.name
#end
```

Downstream call

```
#getRelationships($doc, true) |
#set ( $dnRels = $returnRels )
#foreach ( $dnRel in $dnRels )
  #set($dnDoc = $dnRel.toDocument)
  $dnDoc.documentKey $dnDoc.name
#end
```

Velocity Content Macros

ADD REUSABLE FORMATTING BLOCKS TO YOUR REPORT



Velocity Content Macros

BUILD MODULAR HTML AND DYNAMIC HEADINGS

- Velocity macros can also be used to build HTML content, allowing you to build your report with modular pieces.
- This macro constructs Word headings using item names and a dynamically determined heading level supplied by the \$hierarchyLevel argument.

```
#formatHeading( $doc $hierarchyLevel $docStatus )
```

```
#macro( formatHeading $doc $hierarchyLevel $docStatus )  
  #if( $hierarchyLevel < 7 )  
    <h$hierarchyLevel>  
      <a name="_API$docID"><span style='mso-bookmark:_API$docID'>$doc.name</span></a>  
    </h$hierarchyLevel>  
  #else  
    <p class=MsoHeading$hierarchyLevel>  
      <a name="_API$docID"><span style='mso-bookmark:_API$docID'>$doc.name</span></a>  
    </p>  
  #end  
#end
```

Embedded Static Images

HOW TO EMBED AN IMAGE LIKE A LOGO INTO YOUR REPORT



Embedding Static Images

ADD STATIC IMAGES SUCH AS COMPANY LOGOS TO YOUR REPORT

- Static images in custom reports are images that will always be present when the report is generated and are not supplied from your Jama instance.
 1. Create a Word Mockup Template with your static image (.png, .jpg, .jpeg) pasted into the correct position.
 2. Follow the steps on slides 11 & 12 to convert this file to .mht format and open it on your text editor.
 3. Locate the base64 encoded block for your image which is typically stored at the bottom and copy this section over to your custom report velocity file.

```
-----=_NextPart_01D506B0.1F52D2B0
Content-Location: file:///C:/D993472D/WordTemplate.fld/image001.jpg
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

/9j/4AAQSkZJRgABAQEA3ADcAAD/2wBDAAIBAQEBAQIBAQECAgICAgQDAgICAgUEBAMEBgUGBgYF
BgYGBwkIBgcJBWYGCAsICQoKCgoKBggLDAsKDAkKCgr/2wBDARICAgICAgUDAwUKBwYHCgoKCgoK
CgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgr/wAARCAArA0QDASIA
```

Embedding Static Images

STATIC IMAGE BASE64 HEADING ADJUSTMENTS

1. Ensure the boundary of the base64 code block matches the boundary of the template file you are utilizing. All the boundaries in the template file need to match. Therefore, replace the boundary of the base64 header to match the boundaries of the template file.

```
MIME-Version: 1.0  
Content-Type: multipart/related;  
boundary="-----=_NextPart_01D506B0.1F52D2B0"
```

```
-----=_NextPart_73S849B0.1F52D2B0  
Content-Location: file:///C:/D993472D/WordTemplate.fld/image001.jpg  
Content-Transfer-Encoding: base64  
Content-Type: image/jpeg
```

Note: For a human readable example you could rename your boundaries Example: "-----=_tiramisu" but keep in mind all boundary references will also require this adjustment.

2. Next update the Content-Location: to Content-ID: and remove the file path it references until only the file name remains.

```
-----=_NextPart_01D506B0.1F52D2B0  
Content-ID: image001.jpg  
Content-Transfer-Encoding: base64  
Content-Type: image/jpeg
```

Embedding Static Images

UPDATE THE EMBEDDED FILE LIST

1. At the very bottom of the file you will find the following <xml> block.
2. Simply add your new file reference to this list matching the format shown.

Example:

<o:File HRef="image001.png"/>

```
<xml xmlns:o="urn:schemas-microsoft-com:office:office">
  <o:MainFile HRef="../WordPDFMaster.htm"/>
  <o:File HRef="image001.jpg"/>
  <o:File HRef="item0001.xml"/>
  <o:File HRef="props002.xml"/>
  <o:File HRef="themedata.thmx"/>
  <o:File HRef="colorschememapping.xml"/>
  #foreach( $file in $fileList)
    $!file
  #end
  <o:File HRef="filelist.xml"/>
</xml>
```

Embedding Static Images

STATIC IMAGE REFERENCE USAGE ADJUSTMENTS

1. Back in your .mht file locate the section in the `<body>` where your image code is embedded (it will look like the example on the right).
2. Adjust the code as follows:
 - Remove `<v:shapetype ...>` opening and closing tags and contained content they hold.
 - In `<v:shape ...>` remove all attributes between this opening tag and **style**.
 - In the `<v:imagedata ...>` tag modify the **src** attribute to have the file name prefaced by **"cid:"**

```
<!--[if gte vml 1]><v:shapetype id="_x0000_t75"
coordsize="21600,21600" o:spt="75" o:preferrelative="t"
path="m@4@5l@4@11@9@11@9@5xe"
filled="f" stroked="f">
<v:stroke joinstyle="miter"/>
<v:formulas>
<v:f eqn="if lineDrawn pixelLineWidth 0"/>
<v:f eqn="sum @0 1 0"/>
<v:f eqn="sum 0 0 @1"/>
<v:f eqn="prod @2 1 2"/>
<v:f eqn="prod @3 21600 pixelWidth"/>
<v:f eqn="prod @3 21600 pixelHeight"/>
<v:f eqn="sum @0 0 1"/>
<v:f eqn="prod @6 1 2"/>
<v:f eqn="prod @7 21600 pixelWidth"/>
<v:f eqn="sum @8 21600 0"/>
<v:f eqn="prod @7 21600 pixelHeight"/>
<v:f eqn="sum @10 21600 0"/>
</v:formulas>
<v:path o:extrusionok="f" gradientshapeok="t" o:connecttype="rect"/>
<o:lock v:ext="edit" aspectratio="t"/>
</v:shapetype><v:shape id="Picture_x0020_843937494" o:spid="_x0000_i1025"
type="#_x0000_t75" style='width:127pt;height:35pt;visibility:visible;
mso-wrap-style:square'>
<v:imagedata src="TrainingExerciseTemplate.fld/image001.png" o:title=""/>
</v:shape><![endif]-->
```

Embedding Static Images

FINAL RESULT WITH A PROPER BASE64 ENCODED IMAGE REFERENCE

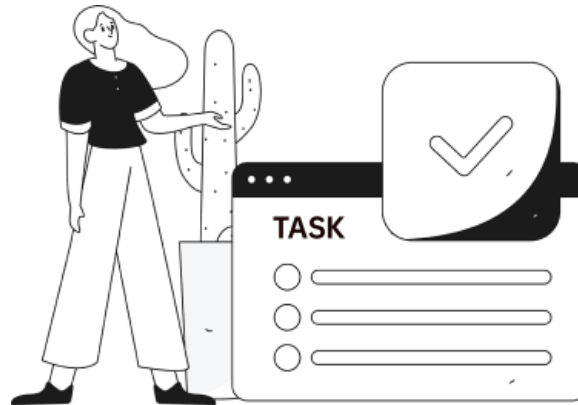
1. Once you are finished with these modifications you should have something that looks like the following where only the `<v:shape>` and `<v:imagedata>` tags are kept and "*cid.*" is injected for the *src* attribute:

```
<!--[if gte vml 1]>
  <v:shape style='width:468pt;height:215pt;visibility:visible;mso-wrap-style:square'>
    <v:imagedata src="cid:image001.jpg" o:title=""/>
  </v:shape>
<![endif]-->
```

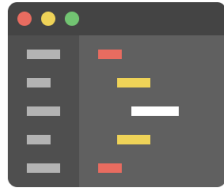
2. This will properly reference your embedded image base64 definition through the "Content-ID" attribute in order to render the image at this location.

Part 1 Close

CODE REFERENCES, WHAT TO EXPECT FOR PART 2, HOMEWORK



Example Code References & Templates



[Jama Community Reports Github Repository](#)

You can utilize the Jama Community Velocity reports as a reference guide of custom reporting. These public reports can provide examples for gathering test runs, reviews, baselines, and more.



[Customer Training Word & Excel Templates](#)

The Jama EESR Team provides you with starter template files for both Word and Excel format reports. These templates include a collection of macros to utilize for formatting, gathering data, and constructing custom reports. These templates are supplied in Part 2 of this training.

HOMework

COMPLETE THESE PROMPTS BEFORE PART 2

- In your provided Part 1 Training materials please find the following files to be used for your Velocity homework exercises:
 - *Basic_Training_Homework_Prompts.docx*
 - *Basic_Training_Mockup_Example.docx*
 - *Basic_Training_Velocity_Template.vm*
- Follow the '*Basic_Training_Homework_Prompts.docx*' so that you can convert the '*Basic_Training_Mockup_Example.docx*' into a .mht file.
- You will then copy parts of this .mht file into the prepared '*Basic_Training_Velocity_Template.vm*' file while its open in your text editor.
- From there complete the prompts in '*Basic_Training_Homework_Prompts.docx*'.