

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

DOCUMENT INFORMATION

Name:	uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3
Number:	D175602
Design History File:	D172870

REVISION HISTORY

Revision (#)	Date	Description
1	See myPLM	Initial Release
2	See myPLM	Added "DIPLOMAT: OpenAPI Digital Pathology Algorithm Output Format" section and renamed to uPath 2.1 Interface Control Document (ICD) for Open API v1.2
3	See myPLM	Updates to Partner Algorithm API, Added Bring Your Algorithm(BYA) details, Added API for Open source libraries

APPROVAL

Name	Signature	Date	Role
Bhanu Prakash	eSignature	See myPLM	Solution Architect
Dmitry Popov	eSignature	See myPLM	Project Management
Karel Zuiderveld	eSignature	See myPLM	Sr.Architect
Jaimini Kamtamneni	eSignature	See myPLM	Engineering Management
Michael Zhou	eSignature	See myPLM	Solution Architect

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

TABLE OF CONTENTS

1.0 PURPOSE	5
2.0 SCOPE	5
3.0 DEFINITIONS	5
4.0 INTERFACE DESCRIPTION	6
Open Environment Architecture	6
Bring Your Platform (BYP) model	6
Bring Your Algorithm (BYA) model	8
BYA Design considerations	9
5.0 SECURITY	10
6.0 DEFECT & INCIDENT MANAGEMENT	10
7.0 PARTNER AND ALGORITHM ONBOARDING	11
8.0 LOCALIZATION	11
9.0 PARTNER API	12
Common Response Structures	12
HTTP Error Response Structure	12
Common Success and Error HTTP Codes	13
9.1 Vendor Details	13
9.2 Algorithm Detail	14
9.3 Algorithms List	30
9.4 Start Analysis (Primary Analysis Request)	31
9.5 Secondary Analysis Request using ROIs(Optional)	33
9.6 Slide Rescore Request using ROIs(Optional)	36
9.7 Stop Analysis	40
9.8 Analysis Status	40
9.9 Close Analysis	42
9.10 Analysis Result	43
9.11 Open Source libraries(Optional)	45
9.12 Submit Analysis Rejection Reason(Not supported)	46
10.0 ROCHE CALLBACK API	47
10.1 Update Analysis Status	47
10.2 Notify Availability (Applicable to BYA only)	48
11.0 AUTHENTICATION AND AUTHORIZATION	49
11.1 Partner API Authentication and Authorization	49
OAuth Access Token Endpoint	49

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

Request Headers	50
11.2 Roche API Authentication and Authorization	51
OAuth Access Token Endpoint	51
12.0 PRIMARY ANALYSIS FLOW INTERACTION DIAGRAM	53
12.1 Single Slide per WSA	53
12.2 Multiple Slides per WSA(Linked slides)	54
13.0 SUPPORTED PATHOLOGIST FLOWS	55
14.0 DIPLOMAT: Roche Open Environment Digital Pathology Algorithm Output Format	56
Goals of this effort	56
Basic assumptions	56
Use-cases that drove DIPLOMAT development	57
Why HDF5?	59
What about DICOM?	60
Roche's Overlay Tile Generator Service (OGS)	60
Contents of a DIPLOMAT HDF5 file	61
wsi_analysis_info: Algorithm and runtime information	62
wsi_analysis_info/diplomat	63
wsi_analysis_info/developer	64
wsi_analysis_info/algorithm	64
wsi_analysis_info/input	66
wsi_analysis_info/platform	68
wsi_analysis_info/runtime	68
wsi_cells: Storage of spatial data (geometry)	69
How to efficiently store GeoJSON data	69
Tile-based GeoJSON	70
Compressed GeoJSON	71
Example: list of cell positions and associated labels	71
Example: list of cell positions, associated labels, and stain intensities	72
Example: Roche Dualish (cell boundaries, black and red dot count)	73
wsi_masks: Storage of segmentation masks	75
Binary masks	75
Single label segmentation masks	76
Multi-label segmentation masks	78
Hybrid multi-label segmentation masks	78
wsi_heatmaps: Heatmaps	79
Heatmap overlays stored as SVG	79
Heatmaps stored as chunked HDF5 tiles	80

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

Heatmaps stored as DICOM	81
wsi_thumbnail: Thumbnail	81
wsi_presentation: Presentation State	82
Support for internationalization: wsi_presentation/locales/...	83
Color specification	85
Color LUT specifications: wsi_presentation/colorluts	86
Marker shape specifications: wsi_presentation/marker_shapes	88
Vertex styles description	89
Rendering wsi_cells contents: wsi_presentation/markers	89
Rendering of masks: wsi_presentation/masks	94
Dynamic rendering of heatmaps: wsi_presentation/heatmaps	96
Displaying panels: wsi_presentation/panels	100
Targeted for future release: annotations and scenes	102
wsi_annotations: Annotations	102
Examples of Flexpoly annotations	103
Specification of cull types using cullTypes.json	107
Dynamic rendering of annotations	108
Putting everything together: scenes	109
wsi_scores: Whole slide analysis scores	112

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

1.0 PURPOSE

The document describes the interface between uPath subsystem and third party digital pathology (DP) image analysis algorithms. uPath will use this interface to perform whole slide analysis (WSA) and related tasks on high resolution tissue images(.bif) of various tissue types supported by the Roche Open Environment algorithm microservice. This document will be shared with partners for API reference and algorithm integration.

2.0 SCOPE

The interface defined in this specification will be consumed by the uPath subsystem and will be implemented by the digital pathology algorithms. This interface will also be able to support any algorithm developed in future which strictly adheres to the workflow for the algorithms listed above.

Roche Open Environment currently supports only bif images generated through Ventana DP200/DP600 scanners.

3.0 DEFINITIONS

Term	Definition
DP	Digital Pathology
uPath subsystem	uPath is a subsystem of the uPath IVD product. It manages workflows and dispatches various user requests to other subsystems.
OpenEnvironment Microservice	Roche Open Environment microservice will be the algorithm workflow layer. This will be the interface for uPath to integrate internal and external algorithms.
AWS S3	AWS S3 for file storage and retrieval. Image Management System.
REST API	Access algorithm over HTTPs
NCR	Negative control slide
PCR	Positive control slide
H&E	H&E (Hematoxylin & Eosin) slide
IHC	Slides with Immunohistochemical staining
WSA	Whole Slide Analysis (also referred as Primary Analysis)
ROI	Region of Interest - an area drawn over the slide for analysis by the pathologist
UUID	Universally unique identifier represented as 32 hexadecimal. Ex.: "b4383140-b915-40b1-aa33-97cff7c13c22"

TEMPLATE: FT0500-504C

All printed copies of this document are uncontrolled – Document Control System contains the most current revision.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

4.0 INTERFACE DESCRIPTION

The Roche Open Environment provides an interface to

- Onboard algorithm vendors and their algorithms to uPath.
- Send images for whole slide analysis / primary analysis
- ROI analysis and secondary analysis of image.
- Analysis life cycle management - start, cancel, delete.
- Handle analysis progress and analysis results.

uPath platform provides numerous functionalities including case accessioning and associate slides to the case, LIS integration, patient information, user roles, slide viewer, algorithm management, 2nd opinion, case sharing, overlay toggling, analysis score etc.

Open Environment Architecture

OpenEnvironment Microservice - REST based service to have an open architecture for algorithm integration and execution.

Internal algorithm - Algorithms developed by Roche and deployed in Roche infrastructure/network.

Bring Your Platform (BYP) - BYP model is developed for integration of algorithms developed by external algorithm partners and deployed in Partner cloud.

Bring Your Algorithm (BYA) - BYA model is developed for integration of algorithms developed and dockerized by external algorithm partners and deployed in Roche infrastructure/network.

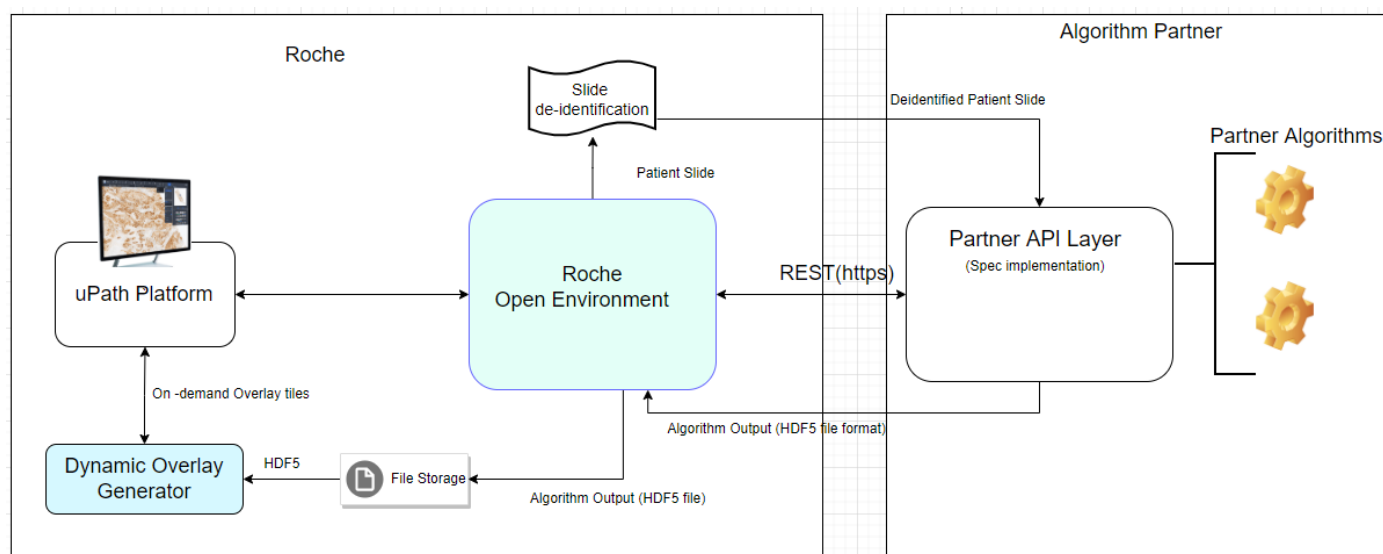
Bring Your Platform (BYP) model

- Partner algorithm to be accessed over HTTPs REST endpoints
- REST based architecture makes it cloud agnostic
- Partner to have access on Roche customer tissue images without PHI
- Images to be accessed by a partner algorithm from AWS S3 bucket in Roche Cloud using a pre-signed url provided through the API.
- Image metadata to be shared with partner and vice versa over REST endpoints
- Algorithm execution and output
 - WSA (Whole Slide Analysis)

TEMPLATE: FT0500-504C

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

- Algorithm output is an HDF5 file with region masks, cell masks and heatmap masks (masks as numeric values).
- uPath Overlay generator module uses the HDF5 file to generate overlay for the whole slide image.
- The score is sent by algorithm using REST API.



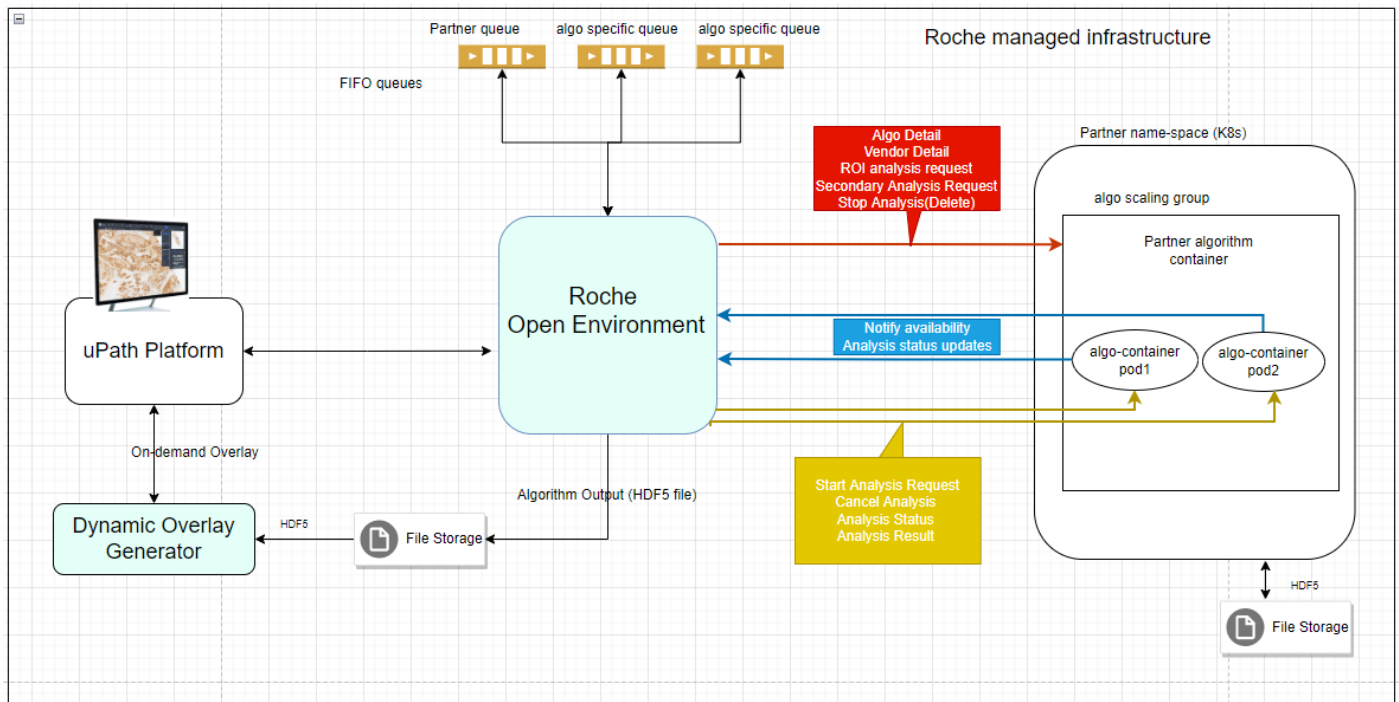
Workflow Steps - BYP

1. uPath calls OpenEnvironment Start Analysis API to start whole slide analysis of an image.
2. OpenEnvironment will invoke the "Start Analysis" partner API to trigger the algorithm execution in the partner cloud.
3. Partner will invoke the "Update Analysis Status" API hosted in OpenEnvironment microservice to send intermediate and final algorithm execution status - INPROGRESS, COMPLETED, FAILED status.
4. OpenEnvironment "Update Analysis Status" endpoint sends a message to uPath to update the status of slide analysis.
5. On completion of algorithm execution, the partner will place the algorithm output as HDF5 file in a AWS S3 bucket and send the COMPLETED status in "Update Analysis Status" API hosted in OpenEnvironment microservice.
6. OpenEnvironment microservice will invoke the "Analysis Result" partner API after getting the COMPLETED status in the "Update Analysis Status" API. "Analysis Result" API response will have a pre-signed AWS S3 url (in partner cloud) which will have the

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

- algorithm output an HDF5 file.
7. OpenEnvironment microservice downloads the HDF5 file from partner S3 bucket and uploads to uPath Tenant S3 bucket, and sends a message to uPath to mark the analysis status as COMPLETED.
8. uPath viewer creates a session with Overlay Generator module which creates the overlay tiles to be displayed on uPath Viewer.

Bring Your Algorithm (BYA) model



Legends :

Blue REST Calls from algorithm instances (pods) to OpenEnvironment

Yellow REST calls from OpenEnvironment to the algorithm instance directly using the IP address of Pod where the algorithm is running.

Red - REST Calls from OpenEnvironment to the scaling group running one or more pods for the same algorithm container. Any of the algorithm instances can receive these calls.

Workflow Steps - BYA

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

1. uPath calls the OpenEnvironment Start Analysis API to start whole slide analysis of an image.
2. Open API will put a message into Algorithm Queue to start analysis. There is a separate queue for each algorithm container. This queue will be used to scale algorithm instances based on unprocessed message count.
3. The algorithm instance shows its availability by calling the /notify endpoint of OpenEnvironment. It keeps pinging this endpoint every 2 minutes until it gets an analysis request from OpenEnvironment.
4. OpenEnvironment checks the queue for any analysis messages. If available, it sends the start-analysis request to the algorithm instance from which it received the /notify call.
5. The algorithm instance starts the analysis and calls OpenEnvironment "Update Analysis Status" endpoint to update interim analysis progress on regular intervals as well as the completion of analysis.
6. Once analysis is completed, the algorithm instance saves the outcome in HDF5 file onto disk with file name <analysisId>.h5 and calls the "Update Analysis Status" endpoint with COMPLETED status.
7. Upon analysis completion, OpenEnvironment sends the request to retrieve the analysis results.
8. Algorithm instance makes itself available for new analysis request by calling the /notify endpoint.
9. uPath viewer creates a session with Overlay Generator module which creates the overlay tiles to be displayed on Viewer.
10. If an algorithm supports the ROI analysis, the OpenEnvironment makes the ROI analysis request by passing the primary(WSA) analysisId. The algorithm shall, if needed, find the HDF5 file for primary analysis from the disk using the analysisId (File name will be <analysisId>.h5).

BYA Design considerations

1. Algorithms shall provide the algorithm information as specified in "Algorithm Detail" API.
2. uPath supports multiple languages. The algorithm shall be capable of providing the strings for different locales.
3. Partner is provided with a directory on disk (EFS volume on AWS or disk) where algorithms can store the analysis output (HDF5 file). There is no other database provided.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

4. An algorithm partner needs to provide the container which will be hosted in Roche cloud and/or on-prem. Following container strategies can be used:
1. **Single-algorithm-per-container** - Each algorithm is provided as a separate docker image.
 2. **One-container-for-multiple-algorithms** - A partner having multiple algorithms can either go with option-1 or provide a single container having all algorithms. Note - Not more than one container are allowed to have multiple algorithms.
 3. **Mix approach** - This approach is helpful when a partner first goes with one-container-for-multiple-algorithms strategy and later wants to bring additional algorithms. The additional algorithms can use only a single-algorithm-per-container approach.

The way the algorithms communicate with Open Environment differ based on the container strategy. Please refer to the [Notify Availability](#) API.

5.0 SECURITY

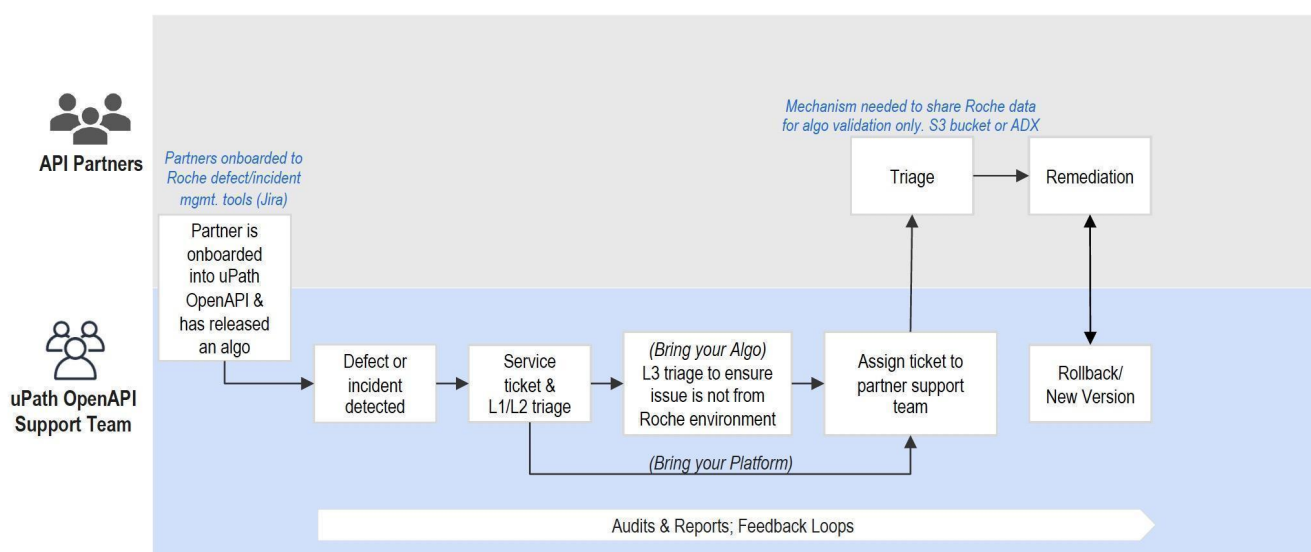
- API security is important when a BYP partner connects to Roche Open Environment.
- All API communication between Roche and the BYP partner needs to be using HTTPS.
- Navify authentication and authorization to be used by Roche APIs. Roche provides the credentials to partners to access its APIs.
- Token to be generated using uPath Navify and sent when making a call to Roche API.
- BYP Partner APIs also need to be secured by partners and credentials need to be shared with Roche.
- All tokens are short lived and need to be refreshed on expiry.
- Token validation for all API calls between Roche and BYP Partner is mandatory before having any exchange of information or accessing resources.
- Penetration testing capabilities & validate there are no critical vulnerabilities prior to Roche Open Environment integration

6.0 DEFECT & INCIDENT MANAGEMENT

- L1 and L2 service through Roche support team
- Only L3 issues to be sent to partner support team

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

Defect & Incident Management



7.0 PARTNER AND ALGORITHM ONBOARDING

A new partner first needs to be onboarded to uPath which establishes the communication between uPath and the partner APIs. A separate jenkins job is used to onboard the partners and their algorithms. A new partner needs to provide following information:

1. Base URL - The base url where REST APIs will be hosted by a partner.
2. Vendor Name - Short name of Partner eg. Roche, PathAI. uPath displays it with the algorithm on various screens and reports.
3. Client Id and secret - to generate the token to call the BYP Partner REST endpoints.

The Onboarding job uses 2 Partner APIs:

1. Vendor Detail - to onboard the partner
2. Algorithm Detail- to onboard and register the Partner algorithms to uPath.

The same job is also used to onboard the new algorithms of an existing Partner.

8.0 LOCALIZATION

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

The uPath application currently supports 21 different languages and may support more locales in future. A list of supported locales is provided below. Algorithm vendors need to maintain and provide the strings as per the locale passed in the endpoint.

- "en" is the default locale.
- The codes (eg. analysis status codes, algorithm parameter keys etc) shall NOT be translated in different languages.
- "Update Analysis Status" callback api shall assume the same locale as provided during the start of the analysis.
- Any error/failure messages from vendor APIs/algorithm shall be as per the locale provided while starting the analysis.

Locales	cs, da, de, el, en, es, fr, hu, it, jp, ko, nl, no, pl, pt, ru, sl, sv, th, tr, zh
----------------	---

A list of all fields to be translated as per the locale is provided below. This list may not be exhaustive and algorithms need to provide translations for any new strings including error messages

API	Strings to be translated
Algorithm Detail	algorithm_description, result_parameters(parameter names, allowed_values_list, prognostic category, classification), tissue type, indication type(if provided).
Algorithms List	Same as Algorithm Detail API
Analysis Status	status detail message
Update Analysis Status(callback)	status_detail_message
Analysis Result	HDF5 presentation state(strings to be displayed on overlay toggle panel, any other strings to be displayed on uPath GUI))

9.0 PARTNER API

Common Response Structures

HTTP Error Response Structure

Error Response	{ "error_code":<int>, "error_message":<string> }
----------------	---

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

--	--

Common Success and Error HTTP Codes

Conventional HTTP response codes and how they are used to indicate success or failure in this API Specification.

HTTP Status Code	Description
200s	Successful request.
400	Payload contains missing or incorrect parameters, specific error messages will be passed using this <u>error response structure</u> .
401	Calling endpoints without passing the authorization token in the request header or invalid token.
403	Calling endpoints that are not accessible under client's permissions.
404	Trying to access an endpoint with an invalid resource identifier. e.g., invalid algorithm id.
405	Calling an endpoint with disabled requests methods. e.g., POST /openapi/v1/algorithms/
429	Making too many API requests too quickly.
500s	Something went wrong on Partner's end.

9.1 Vendor Details

Name	Vendor Details	
Description	Provides information about the algorithm vendor	
Applicable Model	BYP & BYA	
Parameters		
Return Value	vendorInformation	information about a vendor
URL	GET - /openapi/v1/vendor	
Response Time	500ms	
Sample Request	N/A	
Sample Response	<pre>"vendor_information": { "company_name": "Roche",</pre>	

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

	<pre> "company_logo": "iVBORw0KGgoAAAANSUHEUgAAAMgAAABtCAYAAADkiS7DAAAVhk1EQVR42u1dCXQV1Rl+ Wlv1WOvS01ZrS49Hra22tq5V22JrbbVWV6vVWu3xuGAIS9hCEkHAjcWKK5uK4gLuC6IiZ E8gIUBIAoEECUswAiGBkIQkJCEBpv9/573" </pre>	
Success Status Code	2xx	
Response Attributes	company_name	vendor name.
	company_logo	jpeg base64 encoded byte array for Company logo

9.2 Algorithm Detail

Name	Algorithm Detail				
Description	Provides information about the algorithm based on algorithm id				
Applicable Model	BYP & BYA				
Parameters	algorithm id - globally unique "algorithm id" in uuid format				
	locale - The endpoint needs to return the strings in the algorithm information as per the locale provided.				
Return Value	json with algorithm information				information about an algorithm
URL	GET - /openapi/v1/algorithms/<algorithm id>?locale="<locale-val>"				
Response Time	1s				
Sample Request	N/A				
Sample Response	<pre> { "algorithm_id": "41d18c4f-94fe-4686-a706-8b6b7eb47282", "algorithm_name": "PD-L1 22C3", "algorithm_description": "Roche PD-L1 algorithm", "algorithm_type": "<"RUO","IVD","IUO">", "version_number": "1.0", "status": "<"Active","Inactive">", "stain_name": "PD-L1", "tissue_types": [{"key": "BREAST","name": "Breast"}, {"key": "LUNG","name": "Lung"}], "indication_types": [{"key": "MELANOMA","name": "Melanoma"}, {"key": "UROTHELIAL_CARCINOMA","name": "Urothelial Carcinoma"}], "vendor": "Roche", "clone_type": ["SP142", "SP263"], "supported_magnification": "40", "supported_mpp_ranges": [[min, max], [min, max],...], "supported_image_formats": ["BIF","TIF", "TIFF", "SVS"], "supported_scanners":["VENTANA DP 200", "VENTANA DP 600", "APERIO"], </pre>				

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```

"required_slide_types": ["PD-L1"],
"roi_analysis_support":false,
"primary_analysis_overlay_display": true,
"provides_primary_analysis_score": true,
"secondary_analysis_support":true,
"secondary_analysis_annotation_type":<"INCLUSION", "EXCLUSION", "BOTH">,
"max_secondary_analysis_allowed": <int>,
"manual_score_mode": <"INCLUSIVE", "EXCLUSIVE">,
"overlay_acceptance_required": true,
"slide_score_acceptance_required": false,
"requires_analysis_rejection_feedback": false,
"provides_prognostic_score":false,
"results_parameters":
[
{
"name": "Average Cell Count (mm2) ",
"key": "AVG_CELL_COUNT",
"data_type": "float",
"primary_display:true,
"slide_level":true,
"min_value":<float>,
"max_value":<float>,
"precision":<float>,
"can_pathologist_override":true,
"suffix": "%",
"prefix": "+",
"allowed_values_list": [],
"prognostic_group":
[
{
"operator": "LT",
"reference_value": "3000",
"reference_val_min": null,
"reference_val_max": null,
"category": "POSITIVE",
"color_hex_code":<string>
},
{
"operator": "BETWEEN",
"reference_value": null,
"reference_val_min": "3000",
"reference_val_max": "5000",
"category": "EQUIVOCAL",
"color_hex_code":<string>
},...
]
},
{
"name": "Prognostic Score",
"key": "PROGNOSTIC_SCORE",
"data_type": "string",
"primary_display": false,
"slide_level":true,

```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

	<pre> "is_qualitative_score":true, }, { "name": "Evaluable Tissue Area", "key": EVAL_TISSUE_AREA, "data_type": "float", "primary_display": false, "slide_level":true, "can_pathologist_override":false }, { "key": "SCREENING_CLASS", "name": "Screening Class", "data_type": "string", "primary_display": false, "slide_level": true, "can_pathologist_override":false, "classifications": [{"class": "RED","class_name": "Likely Cancer"}, {"class": "YELLOW","class_name": "Likely Benign"}, {"class": "GREEN","class_name": "Undetermined"}] }...] } </pre>				
Success Status Code	2xx				
Response Attributes	Field Name	Type	Optional field?	Default value, if optional	Description
	algorithm_id	string-uu id	No		<p>Partner generated unique identifier for an algorithm(32 hexa + 4 hyphens uuid).</p> <p>Partners shall generate new algorithm id if an upgrade is made to algorithm. In order to maintain the older version of the algorithm, uPath registers the upgraded algorithm as new. Partners shall keep the old algorithm 'Active' if a tenant wants to use it.</p>
	algorithm_name	string	No		String value with a

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

					maximum of 50 characters containing the algorithm name.
	algorithm_description	string	Yes	algorithm_name	String value describing the algorithm. Max 100 char.
	algorithm_type	string	No		Regulatory status of algorithm. Enum value describing whether the algorithm is "RUO", "IVD", or "IUO".
	roi_analysis_support	boolean	Yes	false	<p>Specific to Roche algorithms which support individual ROI scoring but can also be used by partner algorithms if the algorithm provides similar behavior.</p> <p>true - support ROI level analysis. false - no support for ROI level analysis.</p> <p>ROI analysis support allows pathologists to draw the ROI over slide and analyze it in single click. After ROI is analyzed, the uPath viewer displays overlay on ROI area and score is generated at ROI level. An aggregate slide score is also generated. User can draw and analyze more ROIs. Each ROI has individual score generated. Aggregate slide score is updated after each ROI analysis. Algorithms supporting this need to implement Slide Rescore Request API</p>
	primary_analysis_is_overlay_display	boolean	Yes	false	Default false. True if uPath need to display the overlay on whole

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

					slide after primary analysis is completed.
	provides_primary_analysis_score	boolean	Yes	false	Default false. True if algorithm provides slide scores after primary analysis is completed. uPath will display the scores in the Slide Score panel.
	manual_score_mode	string	Yes	EXCLUSIVE	<p>Applicable only if an algorithm allows to override one or more parameter scores (which means at least one algorithm parameter has can_pathologist_override = true).</p> <p>Enum value - either INCLUSIVE or EXCLUSIVE.</p> <p>INCLUSIVE - If pathologist overrides, uPath still displays all the algorithm parameters with their scores - those which are overridden as well as those which are non-editable. The non-editable parameters are displayed with the algorithm provided score.</p> <p>EXCLUSIVE - in this mode, uPath displays only the parameters which are overridden by pathologist. Scores for non-editable parameters are discarded.</p> <p>Note - if pathologist doesn't override any score then this field has no impact and uPath displays the scores as provided by algorithm.</p>
	stain_name	string	No		Enum value with a

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

					maximum of 20 characters describing the stain used to train the algorithm. Examples: "PD-L1", "CD8", "HER2", "Ki-67", "HER2 DUAL ISH", "H&E", "ER", "PR".
	tissue_types	list	No		List of tissue types (key and name pairs) supported by algorithm with a maximum of 20 characters describing the tissue type supported by algorithm. Keys for some common tissue types that partners can use: "PROSTATE" "LUNG" "HEMATOLOGICAL" "BLADDER" "COLORECTAL" "BREAST" "GASTRIC" "SKIN" "LIVER" "LYMPH"
	clone_type	list of strings	Yes	null	List of string value with a maximum of 20 characters describing the clone types used to train the algorithm. eg. ["SP239"]. Note - Currently uPath supports only one clone type for an algorithm.
	indication_types	list	Yes	null	Optional. Indication type or tumor type having max length of 35 characters. Indication Type is currently not supported in uPath.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

	vendor	string	No		Owner of algorithm (same as mentioned in <u>Vendor Detail</u> API)
	supported_scanners	list of strings	No		List of string enums with supported scanners names. "VENTANA DP 200", "VENTANA DP 600", "LEICA", "HAMAMATSU", "APERIO", "ISCAN HT", "ISCAN COREO" Note - Currently uPath Open Environment supports only .bif format using "VENTANA DP 200" and "VENTANA DP 600" scanners.
	supported_image_formats	list of strings	No		List of string enums with supported slide file extensions. Note - Currently uPath Open Environment supports only .bif format.
	supported_magnification	string	Yes if supported_mpp_ranges provides	null	Slide magnification supported by algorithm. Example: "40". Algorithms need to provide either supported_magnification or supported_mpp_ranges.
	supported_mpp_ranges	see example	Yes if supported_magnification provided	null	Contains list of ranges with supported mpp ranges, e.g., [[0.46, 0.54], [0.23, 0.27]] Algorithms need to provide either supported_magnification or supported_mpp_ranges.
	status	string	Yes	"Active"	"Active" or "Inactive"
	version_number	string	No		Algorithm version number (string format). Two types of version formats

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

					are supported: "n.n" eg. "1.0" and "n.n.n" eg. "1.1.0"
	secondary_analysis_support	boolean	Yes	false	Default false. true if algorithm supports secondary analysis. Its needed if pathologist doesn't agree to primary analysis overlay/results. In case of secondary analysis, the pathologist can draw multiple ROIs over the slide and send the slide for re-analysis. The algorithms re-analyses it focusing on areas marked by ROIs and provides a new HDF5 file for overlay as well as the slide score. Algorithms that support secondary analysis, need to implement Secondary Analysis Request API
	max_secondary_analysis_allowed	int	Yes if secondary_analysis_support is false.	0	Default 0. Number of secondary analysis a pathologist is allowed to perform. Applicable if secondary_analysis_support is true.
	secondary_analysis_annotation_type	string	Yes	"BOTH"	Not supported in v1.3. Enum with allowed values - "INCLUSION", "EXCLUSION", "BOTH". Only applicable if algorithm supports secondary analysis. Enum value for type of annotations the algorithm can analyze. Most of the algorithms would have default "BOTH".

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

					<p>INCLUSION - If algorithm supports analyzing the Inclusion annotations only. An "Inclusion" is a region of interest (ROI) that pathologist draws over the slide. Only the area covered under one or more Inclusions is analyzed by algorithm.</p> <p>EXCLUSION - If algorithm needs "Exclusion" areas only. In such case, algorithm will re-analyze all the area of slide excluding the area covered under one or more Exclusions.</p> <p>BOTH - Default value. Area covered by one or more Inclusions minus area intersected by Exclusions. If pathologist chooses to draw no Inclusions on slide and sends only Exclusions then algorithm shall analyze whole slide area minus Exclusions.</p> <p>See Secondary Analysis See ROI Analysis</p>
	required_slide_types	list of strings	Yes, if algo expects only one slide (IHC slide) and doesn't need a linked slide.	stain_name	<p>List of types of slides the algorithm requires to start the analysis. For algorithms which need only one slide, there will be only one value which needs to be the same as the stain_name, eg. [PD-L1]. For algorithms which need to analyze multiple slides together (eg. 1 IHC and 1 Negative Control slide), there will be 2 values - stain_name and type of</p>

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

					slide to be linked to IHC, for example: ["CD8", "NCR"]
	overlay_acceptance_required	boolean	Yes	false	Default false. true if pathologists need to accept the overlay before the slide analysis scores are shown to him.
	slide_score_acceptance_required	boolean	Yes	false	Default false. true if pathologists need to accept the slide analysis score. overlay_acceptance_required will be false if this is true.
	requires_analysis_rejection_feedback	boolean	Yes	false	Optional flag. Not supported currently. Its used by uPath to send the analysis rejection reason provided by pathologist back to algorithm in case an algorithm requires accept/reject of analysis (overlay or score) by the pathologist. The Partner is required to implement the " <u>Submit Analysis Rejection Reason</u> " API if this flag is true.
	provides_prognostic_score	boolean	Yes	false	Default false, true if algorithm provides prognostic reference range for qualitative result. If true, algorithms also need to provide: <ol style="list-style-type: none"> 1. prognostic_group values for the 'key' parameter. 2. Following additional algo parameter for the qualitative score of the slide. <pre>{</pre>

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

					<pre>"name": "Prognostic Score", "key": "PROGNOSTIC_SCORE", "data_type": "string", "primary_display": false, "slide_level": true, "is_qualitative_score": true }</pre>
	results_parameters		No		<p>Contains a list of JSON objects, where each JSON object describes the algorithm parameters for which scores will be generated. Please refer to the following attributes for score parameters.</p> <p>Note - some of the fields are only applicable to the 'key' parameter (parameter having primary_display set to true).</p>
	name	string	No		parameter's name, max 50 character length
	key	string	No		parameter key, max 50 characters (scores are sent by algorithm using parameter keys. Refer Analysis Result API)
	data_type	string	No		<p>Enum "int", "float" or "string".</p> <p>Represents the type of score provided by the algorithm in the Analysis Result. For example - if an algorithm provides a value of 3.5 for a parameter score, the data_type for that parameter needs to be "float".</p>
	primary_display	boolean	No		true if its key

TEMPLATE: FT0500-504C

All printed copies of this document are uncontrolled – Document Control System contains the most current revision.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

					parameter. One parameter for the algorithm needs to be designated for primary_display which is considered as the primary or key parameter and represents key score. For all other parameters, it should be false.
	slide_level	boolean	Yes	true	true if this parameter provides a slide level score. false if it provides only an ROI level score.
	min_value	int or float	Yes		Optional, it is used for input validation in case of manual overriding of score and should only be provided for a parameter for which score can be manually overridden by pathologist (parameter having can_pathologist_override true). Not needed for other parameters. If provided, uPath will not allow the pathologist to enter a value less than the min_value for this score. Not applicable to string parameter scores.
	max_value	int or float	Yes		Optional, it is used for input validation in case of manual overriding of score and should only be provided for a parameter for which score can be manually overridden by pathologist (parameter having can_pathologist_override true). Not needed for other parameters. If provided, uPath will not

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

					allow the pathologist to enter a value more than the max_value for this score. Not applicable to string parameter scores.
	precision	float or int	Yes		Optional, it is used for input validation in case of manual overriding of score and should only be provided for a parameter for which score can be manually overridden by pathologist (parameter having can_pathologist_override true). Not needed for other parameters. If provided, uPath will validate that the score provided by the pathologist is a multiple of this. If a parameter has a score of minimum 5.0 and precision is 0.5 then allowable values will be 5.5, 6.0, 6.5 etc. If user tries to enter 5.2, it will be invalid and user cannot save that. Not applicable to string parameter scores.
	can_pathologist_override	boolean	Yes	false	true - if the algorithm allows editing/manually overriding of score of this parameter by pathologist. false, otherwise.
	is_qualitative_score	boolean	Yes	false	Optional. Needed only if the algorithm provides a qualitative/prognostic score. Its value shall be true only for the PROGNOSTIC_SCORE

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

					parameter. The algorithm shall provide a quantitative score for the slide in this parameter. Refer provides_prognostic_grouping field for more details.
	prefix	string	Yes		Optional, applicable to key parameter only(parameter having primary_display=true). If available, key parameter score will be prefixed with it. Eg. if prefix is "+" and score is 2 then it will be displayed as +2.
	suffix	string	Yes		Optional, applicable to key parameter only(parameter having primary_display=true). If available, key parameter score will be suffixed with it. Eg. if suffix is "+" and score is 2 then it will be displayed as 2+.
	allowed_values_list	list of strings	Yes		Set of allowed values a parameter can have as its score. Supported only if data_type of parameter is INT or STRING. In case of manual override of score, only the values specified in the list_values will be allowed as score for that parameter.
	classifications	json	Yes	null	Optional if the algorithm doesn't provide any classifications for the slide. Only one parameter is allowed to be treated as

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

					<p>classification.</p> <p>The classifications in uPath are represented at slide and case level using color categorization. uPath uses a set of 3 fixed colors - RED, YELLOW, GREEN. The algorithm shall map the classification with these colors. The color categorization is useful for pathologists to prioritize a case, it also makes the slide stand out from other slides. Example: classifications:</p> <pre>[{"class": "RED", "class_name": "Likely Cancer"}, {"class": "YELLOW", "class_name": "Benign"}]</pre> <p>The score for the classification parameter in the Analysis Result API shall be sent by algorithm exactly as provided by algorithm in "class_name".</p> <p>SCREENING_CLASS: "Likely Cancer"</p>
	prognostic_group	json	Yes	null	<p>Optional if an algorithm does not provide prognostic grouping (provides_prognostic_grouping is false). Applicable to key parameter only (parameter having primary_display=true). See following attributes.</p>

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

	operator	string	No if algo provides prognostic grouping		One of the following operators: LT, LTE, EQ, GTE, GT, BETWEEN.
	reference_value	string	Yes		Either reference_value or (reference_val_min and reference_val_max) need to be provided if algo supports prognostic grouping.
	reference_val_min	string	Yes		Either reference_value or (reference_val_min and reference_val_max) need to be provided if algo supports prognostic grouping.
	reference_val_max	string	Yes		Either reference_value or (reference_val_min and reference_val_max) need to be provided if algo supports prognostic grouping.
	category		No if algo provides prognostic grouping		Qualitative value eg. NEGATIVE, POSITIVE etc.
	color_hex_code		No if algo provides prognostic grouping		String value representing Hex color code for a prognostic category.
Request Validation					
Notes					

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

9.3 Algorithms List

Name	Algorithms List
Description	Retrieves the list of partner algorithms to be registered with uPath
Applicable Model	BYP & BYA
Parameters	locale - The endpoint needs to return the strings in the algorithm information as per the locale provided.
URL	GET /openapi/v1/algorithms?locale="<locale-val>"
Response Time	1s
Sample Request	N/A
Sample Response	<pre>[{ "algorithm_id": "41d18c4f-94fe-4686-a706-8b6b7eb47282", "algorithm_name": "PD-L1 (SP263) NSCLC IVD", "algorithm_type": "<"RUO","IVD">", "stain_name": "PD-L1", "vendor": <"Roche","PathAI","FMI">, "version_number": "1.0", "status": <"Active", "In-Active">, }, { "algorithm_id": "138a5aa7-3bb5-49d8-86c3-8b8b075bdc07", "algorithm_name": "HER2 (4B5) Breast RUO", "algorithm_type": "<"RUO","IVD">", "stain_name": "HER2", "vendor": <"Roche","PathAI","FMI">, "version_number": "1.0", "status": <"Active", "In-Active">, },]</pre>
Success Status Code	2xx
Response Attributes	See Algorithm Detail above
Request Validation	N/A
Notes	

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

9.4 Start Analysis (Primary Analysis Request)

Name	Start Analysis	
Description	Initiate a WSA for a given image	
Applicable Model	BYP & BYA	
Parameters	algorithm_id	Partner generated unique identifier for an algorithm.
	image_url	https link to access the whole slide image file. For cloud, it will be presigned S3 url.
	md5	MD5 hash generated using the slide image file in "image_url". This will be used to verify the integrity of the image file.
	slide_magnification	Int or float value(in string representation, example "20", "20.0") capturing the slide's maximum objective power.
	scanner_name	Enum string value containing the scanner's name used to scan the slide.
	microns_per_pixel_x	Float value for the slide's micron-per-pixel in the X dimension. (scan resolution).
	microns_per_pixel_y	Float value for the slide's micron-per-pixel in the Y dimension. (scan resolution).
	slide_width	Integer value for slide's pixel width.
	slide_height	Integer value for slide's pixel height.
	stain	String value for slide's stain (bio-marker). (Refer stain_name field in Algorithm Details API)
	tissue_type	"KEY" for the slide's tissue type. (Refer Algorithm Details API)
	clone_type	String value for slide's clone type. (Refer Algorithm Details API)
	slide_group_id	Optional if the algorithm doesn't need a linked slide for analysis. Globally unique identifier to identify the slide group. A new uuid will be provided each time the slide is sent for primary analysis. The slides associated together will have the same slide_group_id value.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

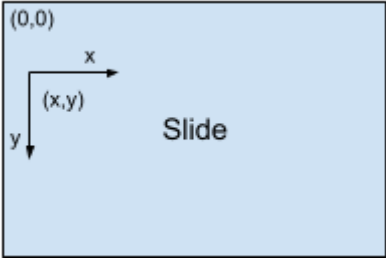
	slide_type	Type of slide NCR - Negative control slide PCR - Positive control slide IHC slide - An IHC slide will have a slide_type value same as the stain_name provided by the algorithm, eg. CD8 or PD-L1.
	indication_type	"KEY" for slide's indication type, if algorithm provides (Refer Algorithm Details API).
	locale	Specifies uPath tenant's locale. Any display strings in results or failure messages for this analysis (including in HDF5 file presentation state) shall be provided as per the locale.
URL	POST /openapi/v1/analysis?locale="<locale-val>"	
Response Time	1s	
Sample	{	

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

Request	<pre>"algorithm_id": "41d18c4f-94fe-4686-a706-8b6b7eb47282", "image_url": <str_link>, "slide_group_id": <uuid>, "slide_type": <"H&E", "HER2", PD-L1>, "md5": <str>, "scanner_name": <"VENTANA DP-200">, "microns_per_pixel_x": <float>, "microns_per_pixel_y": <float>, "slide_magnification": <string>, "slide_width": <int>, "slide_height": <int>, "stain": <enum>, "clone_type": "SP142", "tissue_type": "BLADDER", "indication_type": "UROTHELIAL_CARCINOMA" }</pre>
Sample Response	<pre>{ "analysis_id": <uuid> }</pre>
Success Status Code	202
Response Attributes	analysis_id - uuid generated by a partner algorithm for this analysis.
Request Validation	<ul style="list-style-type: none"> • Checks for valid algorithm_id belonging to the request user. • Checks for valid clone_type are compatible with the algorithm's clone_type. • Checks for valid image_url (valid protocols: http/s). • Image's file extension will be verified against the algorithm's supported file formats. • Checks for whether scanner_name is within the algorithm's supported scanners list. • Checks for valid stain name. • Checks for valid tissue_type. • Checks for whether indication(if provided by algorithm) is within the algorithm's supported indication list. • Checks for whether micron_per_pixel values are within the algorithm's supported micron_per_pixel ranges. • The relative difference between microns-per-pixel x and y cannot vary by more than 0.01. • If a provided slide_group_id is already associated with an existing WSA, checks that the algorithm_id in the request matches the algorithm_id of the existing WSA. • Checks that the scanner_name matches the scanner name of all other WSAs associated with the same slide_group_id (eg. VENTANA DP-200). • Note that additional slide validation is performed asynchronously and appropriate errors will be returned as part of the <u>Update Analysis Status</u> endpoint.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

9.5 Secondary Analysis Request using ROIs (Optional)

Name	Secondary Analysis Request	
Description	For algorithms which support secondary analysis for slides. If a pathologist doesn't agree with Primary Analysis overlay, he/she can draw one or more Regions of Interest (ROIs) and request secondary analysis. Number of max allowed secondary analysis is defined by the algorithm. Secondary analysis provides a new output H5 file and slide-level scores. New H5 file need to have overlay data for ROI areas only.	
Applicable Model	BYP & BYA	
Parameters	primary_analysis_id	Unique analysis_id for the successful Whole Slide Analysis of this slide (sent earlier by Partner as a response to start-analysis API for this slide).
	type	All shapes are sent as Polygon to the algorithm. uPath Viewer tool has the following shapes for drawing the ROIs: RECTANGLE, POLYGON, FREEHAND, ELLIPSE.
	coordinates	<p>List of JSON objects with (x,y) coordinates that reference the polygon's vertices.</p> <p>X,Y Coordinate System</p> <ul style="list-style-type: none"> • X,Y measure pixel dimensions • Always measured on slide's highest resolution • (0,0) is slide's top-left corner • X value increments moving right • Y value increments moving down • X and Y values will be ints 
	is_exclusion	If false, it's an inclusion region. A true value represents an exclusion region.
	locale	Specifies uPath tenant's locale. Any Strings in results or failure messages for this analysis (including in HDF5 file) shall be provided as per the locale.

TEMPLATE: FT0500-504C

All printed copies of this document are uncontrolled – Document Control System contains the most current revision.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

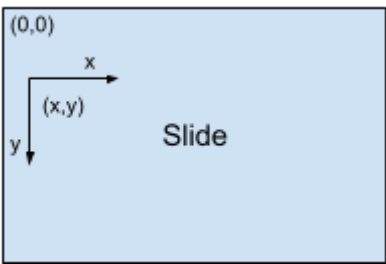
URL	POST - /openapi/v1/analysis/<primary analysis id>/secondary?locale="<locale-val>"
Response Time	<TBD>
Sample Request	<pre>{ "regions":[{ "roi_id":"<string>", "artifacts":[{ "type":"Polygon", "coordinates": [{"x": 6452, "y": 2184}, {"x": 8955, "y": 2184}, ...], "is_exclusion":false }, { "type":"Polygon", "coordinates": [{"x": 6452, "y": 2184}, ...], "is_exclusion":true }, { "type":"Polygon", "coordinates": [{"x": 6452, "y": 2184}, ...], "is_exclusion":true }] }, { "roi_id":"<string>", "artifacts":[{ "type":"Polygon", "coordinates": [{"x": 6452, "y": 2184}, ...], "is_exclusion":false }, { "type":"Polygon", "coordinates": [{"x": 6452, "y": 2184}, ...], "is_exclusion":true }] }] }</pre>

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

	<pre>] } </pre>	
Sample Response	<pre> { "analysis_id":<uuid> } </pre>	
Success Status Code	2xx	
Response Attributes	analysis_id	Partner generated unique identifier for this secondary analysis request
Request Validation	<ul style="list-style-type: none"> • Checks if Primary Analysis for the given analysis id has status "Completed". Any other status will throw an error. • At least one inclusion or exclusion region is required for a secondary analysis request. • There can be multiple inclusions and exclusions in a secondary analysis request. • When a secondary analysis request has both inclusions as well as exclusions, regions of exclusion or the sections of them that fall outside the inclusion region have no effect. The algorithm shall analyze the area under the inclusion regions minus area intersected by exclusion regions. • If the secondary analysis request has only exclusion regions and no inclusion regions, the algorithm shall consider the whole slide area as inclusion region. The analysis shall be done on the whole slide area minus exclusion regions. • Region coordinates form a valid polygon shape, with valid coordinates that are within the slide's bounds. • Each region must have at least three pairs of coordinates. • Checks that all annotations are inside the boundaries of the slide image 	
Notes	<ul style="list-style-type: none"> • Each object in the "regions" list corresponds to a region of interest (ROI). • The order of operations for a request containing multiple ROIs is: <ol style="list-style-type: none"> 1. Within each ROI, remove all of the exclusion regions from the inclusion region. What remains is a region unioned with all the other resultant ROIs. 2. Add all of the resulting regions of each ROI from the previous step together. This aggregate region is what will be analyzed by the algorithm. 3. Area of the slide which falls outside the aggregate ROI is not analyzed. 	

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

9.6 Slide Rescore Request using ROIs (Optional)

Name	Slide Rescore Request (ROI Analysis)	
Description	<p>Analysis request specific to a region of interest. Unlike Secondary Analysis request, it returns the analysis result synchronously (aggregate as well as any ROI level scores) as response. No new HDF5 file is created by algorithm, instead uPath uses primary analysis HDF5 to display the overlay on ROI areas.</p> <p>Note - this endpoint sends one or more ROIs selected by the user. The algorithm is supposed to return the response having aggregate score as well as ROI score for all the ROIs in the request(if algorithm provides ROI level scores too).</p>	
Applicable Model	BYP & BYA	
Parameters	primary_analysis_id	Partner generated unique analysis_id for the successful Whole Slide Analysis of this slide(sent earlier by Partner as a response to start-analysis API for this slide).
	type	All shapes are sent as Polygon to the algorithm. uPath Viewer tool has the following shapes for drawing the ROIs: RECTANGLE, POLYGON, FREEHAND, ELLIPSE.
	coordinates	<p>List of JSON objects with (x,y) coordinates that reference the polygon's vertices.</p> <p>X,Y Coordinate System</p> <ul style="list-style-type: none"> • X,Y measure pixel dimensions • Always measured on slide's highest resolution • (0,0) is slide's top-left corner • X value increments moving right • Y value increments moving down • X and Y values will be ints 
	is_exclusion	If false, it's an inclusion region. A true value represents an exclusion region. An ROI can have max 1 inclusion and 0 or more exclusions.
	locale	Specifies uPath tenant's locale. Any Strings in

TEMPLATE: FT0500-504C

All printed copies of this document are uncontrolled – Document Control System contains the most current revision.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

		results or failure messages for this analysis shall be provided as per the locale.
URL	POST - /openapi/v1/analysis/<primary analysis id>/roi?locale="<locale-val>"	
Response Time	1s	
Sample Request	<pre>{ "regions": [{ "roi_id": "<string>", "artifacts": [{ "type": "Polygon", "coordinates": [{"x": 6452, "y": 2184}, {"x": 8955, "y": 2184}, ...], "is_exclusion": false }, { "type": "Polygon", "coordinates": [{"x": 6452, "y": 2184}, ...], "is_exclusion": true }, { "type": "Polygon", "coordinates": [{"x": 6452, "y": 2184}, ...], "is_exclusion": true }] }, { "roi_id": "<string>", "artifacts": [{ "type": "Polygon", "coordinates": [{"x": 6452, "y": 2184}, ...], "is_exclusion": false }, { "type": "Polygon", "coordinates": [{"x": 6452, "y": 2184}, ...], "is_exclusion": true }] }] }</pre>	

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

	<pre>] }] } </pre>	
Sample Response	<pre> { "aggregate_score": { "EVAL_TISSUE_AREA": 50.5, "TOTAL_TUMOR_CELLS": 60000 }, "roi_scores": [{ "roi_id": "<string>", "TOTAL_TUMOR_CELLS": 10000, "EVAL_TISSUE_AREA": 20.5 }, { "roi_id": "<string>", "TOTAL_TUMOR_CELLS": 50000, "EVAL_TISSUE_AREA": 30.0 }] } </pre>	
Success Status Code	2xx	
Response Attributes	Result JSON having scores for algorithm parameters.	The json needs to have parameter keys as provided in "Algorithm Detail" API. Parameter key scores outside the "roi_scores" in JSON represent the aggregate score. ROI level scores are placed inside the "roi_scores" list. roi_scores is optional and will be used by the algorithms which provide ROI level scores too.
Request Validation	<ul style="list-style-type: none"> Checks if Primary Analysis for the given analysis id has status "Completed". Any other status will throw an error. One inclusion per ROI is required. Exclusions are optional. There can be multiple exclusions per ROI. Region of exclusion or the sections of them that fall outside the inclusion region have no effect. There can be multiple ROIs with each ROI having one mandatory inclusion and optionally one or more exclusions. Region coordinates form a valid polygon shape, with valid coordinates that are within the slide's bounds. 	
Notes	<ul style="list-style-type: none"> Each object in the "regions" list corresponds to a region of interest (ROI). 	

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

- The order of operations for a request containing multiple ROIs is:
 1. Within each ROI, remove all of the exclusion regions from the inclusion region. What remains is a region to be analyzed for the ROI.
 2. Add all of the resulting regions of each ROI from the previous step together. This aggregate region is what will be analyzed by the algorithm.
 3. Area of the slide which falls outside the resulting regions is not analyzed.
 4. While calculating the aggregate score, algorithms shall take into consideration that there may be overlapping area between 2 ROIs. For example - cell count for overlapping area shouldn't be added twice in the aggregate score.

9.7 Stop Analysis

Name	Stop Analysis	
Description	Stops an in-progress WSA.	
Applicable Model	BYP & BYA	
Parameters	analysis_id(UUID)	Unique identifier referencing the WSA.
Response Time	1s	
URL	POST - /openapi/v1/analysis/<analysis id>/stop	
Sample Request	N/A	
Sample Response	See Analysis Status response	
Success Status Code	2xx	
Response Attributes	See Analysis Status	
Request Validation	<ul style="list-style-type: none"> • Will return an error message when analysis status is not "Pending", "In-progress" or "Accepted". 	
Notes	<ul style="list-style-type: none"> • "Cancelled" status to be returned in response • In case an algorithm requires linked slides(eg. 1 IHC and 1 NCR), the stop-analysis request will be sent only for the analysis id belonging to stained(IHC) slide. Partners need to make sure that they update the status of the linked slide to "Cancelled" too. 	

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

9.8 Analysis Status

Name	Analysis Status	
Description	Provides percentage completion and status of the image analysis for the specified image.	
Applicable Model	BYP & BYA	
Parameters	analysis_id	Unique identifier referencing the WSA.
	locale	any error message shall be returned in locale specific language.
Response Time	1s	
URL	GET /openapi/v1/analysis/<analysis_id>/status?locale="<locale-val>"	
Sample Request	N/A	
Sample Response	<pre>{ "analysis_id": "41d18c4f-94fe-4686-a706-8b6b7eb47282", "started_timestamp": <datetime>, "last_updated_timestamp": <datetime>, "status": < "Pending", "Accepted", "In-progress", "Closed", "Completed", "Failed", "Cancelled">, "percentage_completed": <int>, "status_detail_message": <str> }</pre>	
Success Status Code	2xx	
Response Attributes	analysis_id	Unique identifier referencing the WSA.
	started_timestamp	UTC timestamp when analysis was initiated. In ISO Format "yyyy-MM-dd'T'HH:mm:ss.SSSSSS'Z'"
	last_updated_timestamp	UTC timestamp when analysis was last updated. When status="Completed", algorithm execution time will be computed from started_timestamp and last_updated_timestamp. In ISO Format "yyyy-MM-dd'T'HH:mm:ss.SSSSSS'Z'"
	status	String enum value contains one of the following. Note - These status values are treated as codes and shall NOT be translated in locale specific languages. 1) "Pending" - Analysis request waiting for data validation or if an algorithm needs

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

		<p>more than one slide to start the analysis and is waiting to receive the other slide (hasn't yet received all expected slides).</p> <p>2) "Accepted" - When an analysis request is accepted by a partner, but not yet started. All validations on analysis request/slide passed. If any validation error occurs, Partner algo shall call Update Analysis Status callback API with 'Failed' status.</p> <p>3) "In-progress" - Analysis request in progress</p> <p>4) "Completed" - Algorithm execution completed status from partner</p> <p>5) "Failed" - On any failure of analysis execution in partner cloud</p> <p>6) "Cancelled" - Response Status for stop analysis request.</p> <p>7) "Closed" - Response Status for delete analysis request</p>
	percentage_completed	Integer number to display the progress percentage on UI. This is only an approximate value. None is returned if the analysis status is Stopped, Closed, or Failed.
	status_detail_message	Optional field to return status message.
Notes	Returns HTTP error 404 if <analysis_id> is incorrect/not found.	

9.9 Close Analysis

Name	Close Analysis	
Description	Closes a WSA. Once closed, the current results for the WSA will be the final results (even if the WSA is still processing). No additional results will be stored and no additional ROI requests can be made against it.	
Applicable Model	BYP & BYA	
Parameters	analysis_id(UUID)	Unique identifier referencing the WSA.(analysis id)
URL	POST - /openapi/v1/analysis/<analysis_id>/close	
Response Time	1s	
Sample Request	N/A	
Sample Response	See <u>Analysis Status</u> response	

TEMPLATE: FT0500-504C

All printed copies of this document are uncontrolled – Document Control System contains the most current revision.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

Success Status Code	202
Response Attributes	See Analysis Status
Notes	<ul style="list-style-type: none"> While this endpoint explicitly closes the primary analysis, the algorithms may close them automatically after 60 days. The algorithm must purge all the slides after 60 days of analysis completion. Returns 'Closed' status in response In case an algorithm requires linked slides(eg. 1 IHC and 1 NCR), the delete-analysis request will be sent only for the analysis id belonging to stained(IHC) slide. Partners need to make sure that they update the status of the linked slide to "Closed" too.

9.10 Analysis Result

Name	Analysis Result		
Description	Get the primary or secondary analysis result based on the analysis_id. If an algorithm needs more than 1 slides per analysis(eg. 1 IHC and 1 Negative control), the response will contain analysis results for all the slides.		
Applicable Model	BYP & BYA		
Parameters	analysis_id		analysis id for which results are to be returned. Note - If the algorithm needs more than 1 slides(some algorithms use 2 slides linked together eg 1 IHC and 1 Negative control slide) to perform analysis, the endpoint will still send only one analysisId in Analysis Result request but the Algorithm shall make sure to return the results for both the linked slides as shown in the sample response.
	locale		Any strings in the HDF5 file presentation state(strings that display on uPath GUI) shall be provided as per the locale. Any error message shall be returned in locale specific language.
Response Time	1s		
URL	GET - /openapi/v1/analysis/<analysis id>/result?locale="<locale-val>"		
Sample Request	N/A		
Sample Response	[{		

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

(single slides per analysis)	<pre> "analysis_id": "<uuid>", "result_file": "<str_link>", "results": <JSON> }] </pre>		
Sample Response (Multiple slides per analysis)	<pre> [{ "analysis_id": "<uuid>", "result_file": "<str_link>", "results": <JSON> }, { "analysis_id": "<uuid>", "result_file": "<str_link>", "results": <JSON> }] </pre>		
Success Status Code	200		
Parameters	No parameters		
Response Attributes	FieldName	Optional field?	Description
	result_file	No	String HTTPS link to access the algorithm output generated as a HDF5 (.h5 file) file. The file needs to have .h5 extension. The file will be at partner's server and will be downloaded by uPath using this url. For partners using cloud(AWS), the URL may be a pre-signed S3 url valid for 15 minutes or more. If the URL expires, re-request the result endpoint.
	analysis_id	No	analysis id associated with the slide.
	results	Conditionally Optional: 1. It is optional for primary analysis (WSA) for algorithms which don't provide primary analysis score for WSA analysis (If	<p>JSON object containing the algorithm specific key-value pairs with result fields and values. The json needs to have parameter keys as provided in "Algorithm Detail" API.</p> <p>eg. "results":<JSON></p> <p>The <JSON> will have following format:</p> <pre> { "ARTIFACT_DETECTED": 13, </pre>

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

		<p>provides_primary_analysis_score field is false in Algorithm Details API). Mandatory if algorithm provides primary analysis score.</p> <p>2. Mandatory for secondary analysis (if algorithm supports secondary analysis - refer field secondary_analysis_supported in Algorithm Details API)</p>	<pre>"TUMOR_AREA_DETECTED": 14, "EVAL_TISSUE_AREA": 50, "TOTAL_TUMOR_CELLS": 60000 }</pre>
Request Validation	<ul style="list-style-type: none"> Returns an error if the WSA is not in a "Completed" state. 		

9.11 Open Source libraries(Optional)

URL	GET /openapi/v1/algorithms/<algorithm_id>/openup?locale="<locale-val>"	
URL Parameters	algorithm_id	Algorithm Id (UUID) for which open source libraries list is requested.
Description	<p>API to get the PDF file for open source library list used by algorithm in base64 encoded byte array format.</p> <p>Legal requirement on uPath product - uPath needs to have a way to disclose the list of all open source libraries used by uPath as well as the algorithms associated with it.</p> <p>Algorithm vendors are required to provide the list for each algorithm in PDF format. The list needs to have the following 3 attributes at minimum - Name of Open source software, major-version, License Type. A recommended table is given below:</p>	

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

	<table> <tr> <th>Open Source Software</th><th>Major Version</th><th>Licenses</th></tr> <tr> <td>com.sun.xml.ws.jaxws-rt</td><td>2.x</td><td>Eclipse Distribution License - v 1.0</td></tr> <tr> <td>com.sun.xml.ws.jaxws-tools</td><td>2.x</td><td>Apache-2.0;Eclipse Distribution License - v 1.0</td></tr> <tr> <td>com.sun.xml.ws.jaxws-tools</td><td>2.x</td><td>Eclipse Distribution License - v 1.0</td></tr> <tr> <td>com.sun.xml.ws.policy</td><td>2.x</td><td>Eclipse Distribution License - v 1.0</td></tr> <tr> <td>com.sun.xml.ws.policy</td><td>2.x</td><td>Eclipse Distribution License - v 1.0</td></tr> <tr> <td>com.sun.xml.ws.release-documentation</td><td>2.x</td><td>Unknown license</td></tr> </table>		Open Source Software	Major Version	Licenses	com.sun.xml.ws.jaxws-rt	2.x	Eclipse Distribution License - v 1.0	com.sun.xml.ws.jaxws-tools	2.x	Apache-2.0;Eclipse Distribution License - v 1.0	com.sun.xml.ws.jaxws-tools	2.x	Eclipse Distribution License - v 1.0	com.sun.xml.ws.policy	2.x	Eclipse Distribution License - v 1.0	com.sun.xml.ws.policy	2.x	Eclipse Distribution License - v 1.0	com.sun.xml.ws.release-documentation	2.x	Unknown license
Open Source Software	Major Version	Licenses																					
com.sun.xml.ws.jaxws-rt	2.x	Eclipse Distribution License - v 1.0																					
com.sun.xml.ws.jaxws-tools	2.x	Apache-2.0;Eclipse Distribution License - v 1.0																					
com.sun.xml.ws.jaxws-tools	2.x	Eclipse Distribution License - v 1.0																					
com.sun.xml.ws.policy	2.x	Eclipse Distribution License - v 1.0																					
com.sun.xml.ws.policy	2.x	Eclipse Distribution License - v 1.0																					
com.sun.xml.ws.release-documentation	2.x	Unknown license																					
Applicable Model	BYP & BYA																						
Sample Request	-																						
Sample Response	<pre>{ "open_source_list_pdf": "iVBORw0KGgoAAAANSUHEUgAAAMgAAABtCAYAAADkiS7DAAAVhk1EQVR42u1dCXQ V1Rl+Wlv1WOvS01ZrS49Hra22tq5V22JrbVW6vVWu3xuGAIS9hCEkHAjcWKK5u K4gLuC6IiZE8giUBIAoEECUswAiGBkIQkJCEBpv9" }</pre>																						
Success Status Code	200																						
Request Parameters	-																						
Response Attributes	open_source_list_pdf	pdf;base64 encoded byte array for the pdf file containing open source libraries list																					

9.12 Submit Analysis Rejection Reason (Not supported)

URL	POST /openapi/v1/analysis/<analysis_id>/reject	
URL Parameters	analysis_id	UUID of the analysis to reject.
Description	If an analysis is rejected by a pathologist, send the rejection reason (text) entered by pathologist back to the algorithm.	
Applicable Model	BYP & BYA	
Sample	{	

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

Request	<pre>"analysis_reject_reason": "The algorithm doesn't seem to handle this particular type of tissue well." }</pre>	
Sample Response	-	
Success Status Code	200	
Request Parameters	analysis_reject_reason	String explaining why the pathologist did not agree with the ROI results generated by the algorithm. 5000 character limit.
Response Attributes	-	
Request Validation	<ul style="list-style-type: none"> Checks if the analysis results are already available (makes no sense to give feedback otherwise). Checks that this endpoint wasn't already successfully invoked on the given analysis ID. Checks that the length of the reason doesn't exceed 5000 characters. 	

10.0 ROCHE CALLBACK API

The domain used for the Roche API will be configurable for each Partner API supplied.

10.1 Update Analysis Status

Name	Update Analysis Status	
Description	Roche Open Environment API. The partner algorithm must call this API to update the intermediate and completion status of analysis.	
Applicable Model	BYP & BYA	
Parameters	analysis_id (UUID)	Identifier for the analysis record to be updated
	status (String)	"status":<"Accepted", "In-progress", "Stopped", "Completed", "Failed", "Cancelled">
	percentage_complete (int)	Analysis progress percentage.
	started_timestamp (datetime)	UTC timestamp when WSA was initiated. In ISO Format "yyyy-MM-dd'T'HH:mm:ss.SSSSSS'Z'"

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

	last_updated_timest amp (datetime)	UTC timestamp when WSA was last updated. In ISO Format "yyyy-MM-dd'T'HH:mm:ss.SSSSSS'Z'"
Response Time	1s	
URL	PUT - /openapi/v1/analysis	
Sample Request	<pre>{ "analysis_id": "41d18c4f-94fe-4686-a706-8b6b7eb47282", "status": "In-progress", "status_detail_message": "Algorithm execution in progress", "percentage_completed": 50, "started_timestamp": "<datetime>", "last_updated_timestamp": "<datetime>" }</pre>	
Sample Response	N/A	
Success Status Code	200	
Response Attributes	200 HTTP status code	
Request Validation		
Notes	<p>An API maintained by Roche to get the analysis status from Partner. Partner will invoke this API multiple times</p> <p>Localization - Algorithms shall provide translated strings for any error messages(status_detail_message) based on the locale value provided while starting the analysis.</p>	

10.2 Notify Availability (Applicable to BYA only)

Name	Roche Open Environment API. The partner algorithm must call it to notify Algorithm's availability/readiness for WSA
Description	<p>Endpoint is called by BYA algorithm running in a container to notify Roche Open Environment module of its availability and readiness to accept new whole slide analysis request.</p> <p>The algorithm container shall make this request when no primary analysis(WSA) is active or current primary analysis is finished so that a new slide can be sent by uPath for analysis.</p> <p>Frequency: 30 seconds. Algorithm container shall keep calling this API at 30 seconds intervals until it receives a new primary analysis request.</p> <p>The way an algorithm notifies the Open Environment differs based on the container strategy used. Please refer to the description of algorithm_id field. Please refer to the BYA Design Considerations section for container strategies.</p>
Applicable Model	BYA

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

Parameters	algorithm_id (UUID)	Algorithm Id(UUID) if the container has only one algorithm (single-algorithm-per-container model). Send null if the container has more than one algorithm (One-container-for-multiple-algorithms model).
	vendor	Vendor name eg. Roche, IBEX, PathAI
	ip_address	Current IP address and port (Example "35.195.102.159:2007") where the container is running. This address will be used by uPath to send the start-analysis request to the algorithm.
URL	POST - /openapi/v1/notify	
Sample Request	{ "algorithm_id":"41d18c4f-94fe-4686-a706-8b6b7eb47282", "vendor":"Roche", "ip_address":"35.195.102.159:2007" }	
Sample Response	N/A	
Success Status Code	200	
Response Attributes	200 HTTP status code	
Request Validation		

11.0 AUTHENTICATION AND AUTHORIZATION

11.1 Partner API Authentication and Authorization

OAuth 2.0 (aka OAuth) authorization framework will be used for authorizing access to the Partner API endpoints. Partner will provide client credentials, client id and client secret that can be used to call an endpoint to retrieve an access token.

Client Credentials Grant OAuth 2 flow can be used to directly exchange the client id and secret to retrieve a short-lived access token described [here](#). This access token needs to be passed in the authorization header for every request, which is described [here](#).

OAuth Access Token Endpoint

URL	POST/oauth2/token
Description	Retrieve an access token using OAuth client credentials grant type to then authorize API requests.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

Response Time	<1s	
Request Header	"Content-Type": "application/x-www-form-urlencoded"	
Request	<pre>{ "grant_type": "client_credentials", "client_id": "<sa_service_client_id>", "client_secret": "<sa_service_client_secret>" }</pre>	
Response	<pre>{ "access_token":<token str>, "token_type": "Bearer", "expires_in": 3600, }</pre>	
Success Status Code	201	
Request Parameters	client_id	Partner issued client id. This value should be stored securely.
	client_secret	Partner issued secret key. This value should be stored securely.
	grant_type	Set this to 'client_credentials', which is the OAuth grant type being used to issue an access token.
Response Attributes	access_token	UUID representing the access token.
	token_type	Always set to 'Bearer'.
	expires_in	Integer value representing the TTL in seconds of the access token. This will be set to 3600s (1 hr)
Request Validation	<ul style="list-style-type: none"> Only `client_credentials` grant type is supported, any other grant type will return an error. Checks for validity of client credentials (client_id and client_secret). 	
Notes	Parameters need to be passed via the message body.	

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

Request Headers

Once an access token is issued, all requests need to pass the access token as part of the authorization request header. Below illustrates an example.

Request Header	Authorization: Bearer {access-token}
Example	<pre>curl -H "Authorization: Bearer qZGb4oQrk9qZGb4oQrk9" \ https://www. Roche .com/openapi/v1/algorithms/1234/</pre>
Validation	All issues associated with the access-token will return a 401 unauthorized error. The error response will contain the error specifics such as invalid token and token expiration.

11.2 Roche API Authentication and Authorization

OAuth Access Token Endpoint

URL	POST /api/v1/auth/protocols/oidc/token
Description	Retrieve an access token using the OAuth client credentials grant type to then authorize API requests. Roche Callback API Update Analysis Status is protected. A token must be generated and provided with each call using this API.
Applicable Model	BYP
Response Time	<1s
Request Header	"Content-Type": "application/x-www-form-urlencoded"
Request Body	<pre>{ "grant_type": "client_credentials", "client_id": "<client_id>", "client_secret": "<client_secret>", "scope": "<scope>" }</pre>
Response	<pre>{ "access_token":<token str>, }</pre>

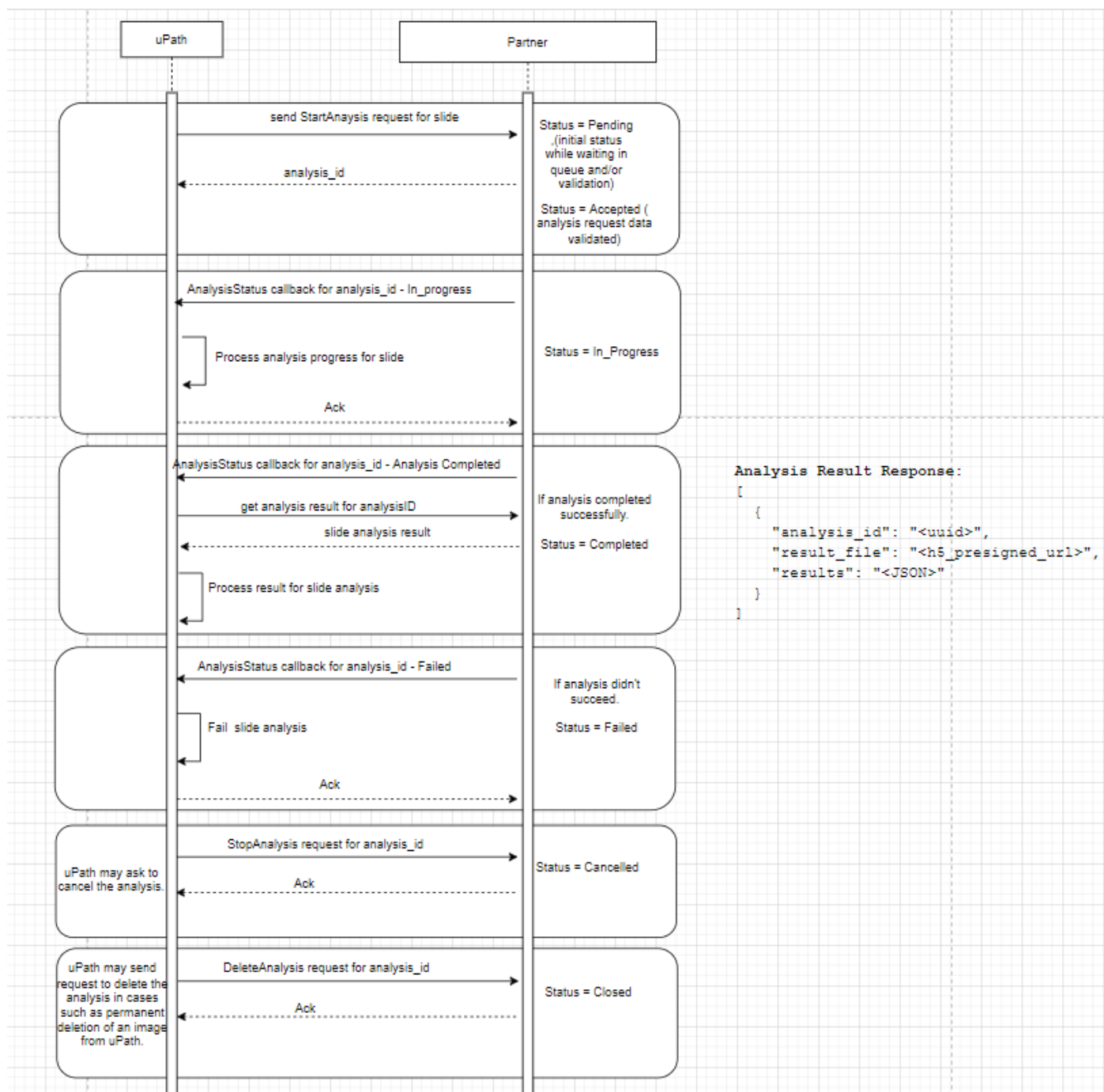
TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

	<pre>"token_type": "bearer", "expires_in": 3600 }</pre>	
Success Status Code	201	
Request Parameters	client_id	Roche issued a client id. This value should be stored securely.
	client_secret	Roche issued secret key. This value should be stored securely.
	grant_type	Set this to 'client_credentials', which is the OAuth grant type being used to issue an access token.
	scope	scope
Response Attributes	access_token	access token.
	token_type	Always set to 'Bearer'.
	expires_in	Integer value representing the TTL in seconds of the access token. This will be set to 3600s (1 hr)
Request Validation	<ul style="list-style-type: none"> Only `client_credentials` grant type is supported; any other grant type will return an error. Checks for validity of client credentials (client_id and client_secret). 	
Notes	Parameters need to be passed via the message body.	

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

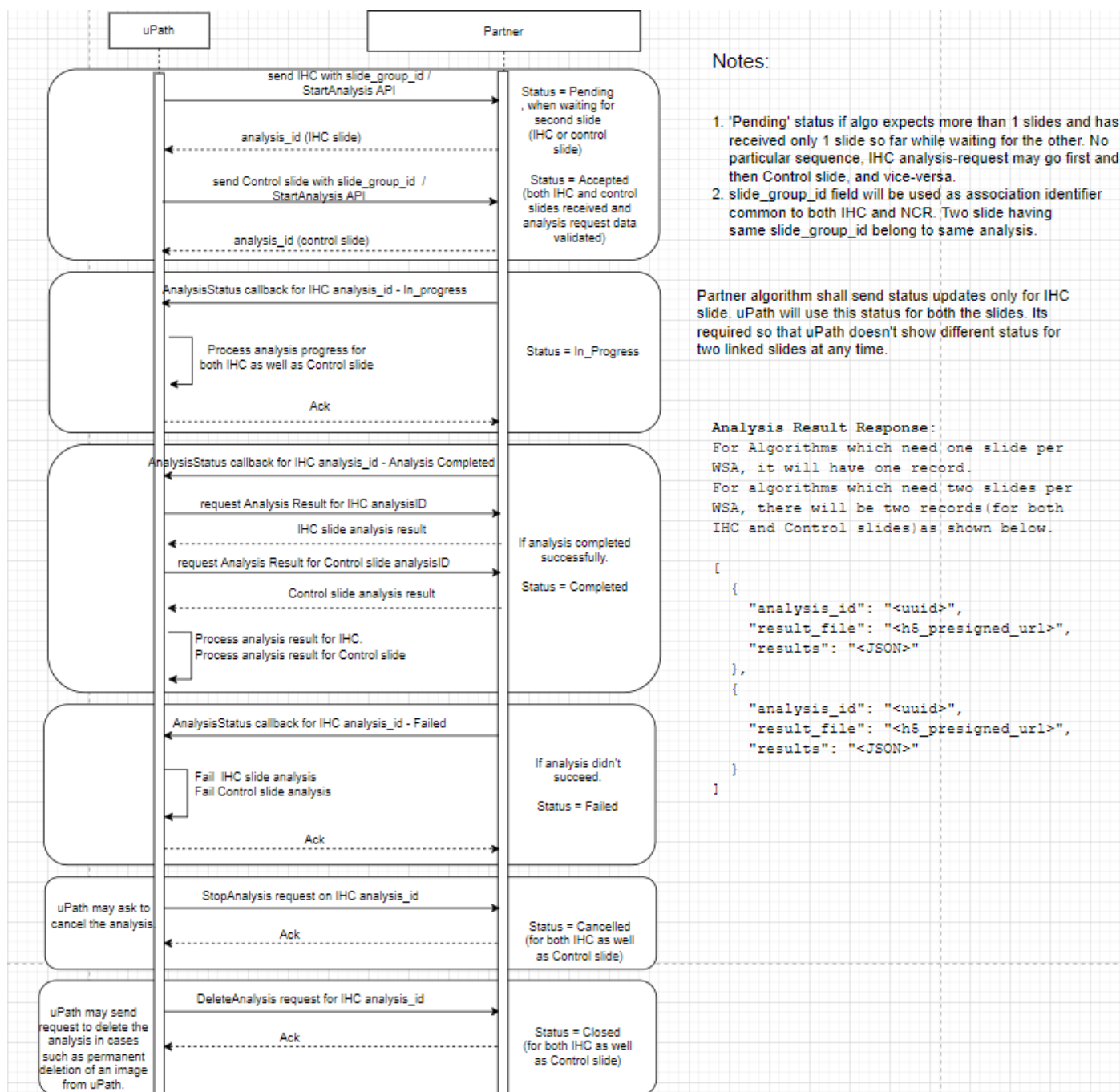
12.0 PRIMARY ANALYSIS FLOW INTERACTION DIAGRAM

12.1 Single Slide per WSA



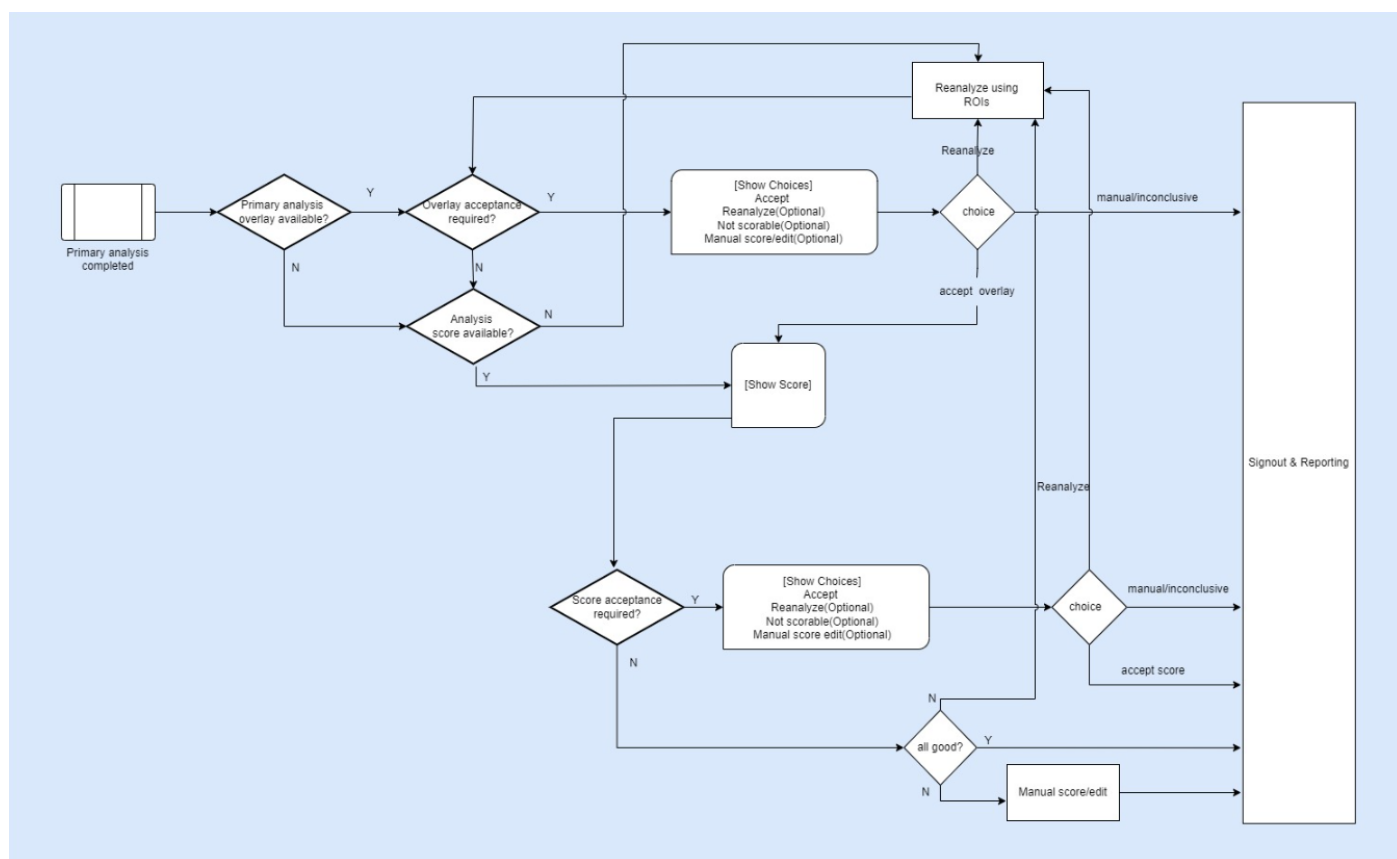
TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

12.2 Multiple Slides per WSA(Linked slides)



TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

13.0 SUPPORTED PATHOLOGIST FLOWS



TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

14.0 DIPLOMAT: Roche Open Environment Digital Pathology Algorithm Output Format

Goals of this effort

Within Roche, there are a plethora of different approaches for storing the output of Whole Slide Analysis (WSA) algorithms; this reduces the overall productivity of our imaging scientists as they often need to deal with platform incompatibilities.

This document describes a common algorithm output to be used within Roche moving forward; in addition, we will share the specification with external Roche collaborators. The new file format is an integral part of the Roche Open Environment 1.3 specification that is used to deploy 3rd-party algorithms on Roche's platforms.

Moving forward, the new format will be referred to as "DIPLOMAT" (DIgital Pathology algorithm Output forMAT), or "Roche Open Environment HDF5". Besides providing a standard format to store the results of WSA algorithms, Roche Open Environment 1.3 DIPLOMAT also provides a standard format for annotations, support for internationalization, presets that specify how algorithm results can be converted into overlays, and a specification of GUI panels that control the display of those overlays.

Basic assumptions

DIPLOMAT uses [HDF5](#) as a data container; this widely-used file format is already used by Roche to store algorithm output masks and has been shown to work well in a production environment. For the new format, we have a few key assumptions and requirements:

1. Wherever desirable, DIPLOMAT is backward-compatible with the current Iris-E HDF5 format used by Roche (documentation accessible via Roche intranet).
2. To maximize utility for imaging scientists, the self-describing DIPLOMAT format should be readable in Python; the format relies heavily on JSON and GeoJSON for storing metadata and algorithm results.
3. The first release of DIPLOMAT should support encoding the output of all current clinical Roche algorithms as well as other algorithms that need to be deployed.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

4. Any feedback and requirements from external partners should be addressed whenever feasible.
5. DIPLOMAT should be convertible to/from the [new DICOM annotation standard](#).

We expect the various groups within Roche to make changes in their tools and products to support the DIPLOMAT format when appropriate. Amongst others, this will include a refactoring of Roche algorithms and platforms to support DIPLOMAT as well as creating new tools like a standalone viewer that can parse and display DIPLOMAT results.

Use-cases that drove DIPLOMAT development

The primary use-case for DIPLOMAT is:

1. An algorithm performs analysis of a whole-slide image
2. The whole-slide analysis results are written to DIPLOMAT
3. DIPLOMAT is used as input for secondary analysis and overlay generation.

Step 3) might be inspection of results using a Python development environment, but also clinical applications. In the latter case, step 3) involves interactive inspection and manipulation of algorithm results, like drawing annotations, selecting appropriate heatmap overlays, and doing secondary analysis (i.e. scoring) of the study.

In clinical studies, Roche applications usually capture user annotations, the corresponding algorithm scores as well as the actual diagnosis from the pathologist which then is used as ground-truth for algorithm development. This leads to another use-case that DIPLOMAT must support:

1. A whole-slide image is analyzed using an algorithm.
2. Output is written to DIPLOMAT.
3. DIPLOMAT files are read by clinical applications; any user annotations used by clinical applications should be convertible into Roche Open Environment/DIPLOMAT JSON format.
4. The whole-slide image is analyzed again, this time using the user annotations (in Roche Open Environment/DIPLOMAT format) as input; the input annotations and generated algorithm scores are stored in the resulting DIPLOMAT file.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

5. The scores in DIPLOMAT are compared against the ground-truth obtained in step 3).

Steps 4 and 5 are repeated during algorithm development until the algorithm satisfies the requirements. This use-case required the development of a standardized annotation format that could be included in the DIPLOMAT standard.

The third important use-case for DIPLOMAT is achieving a form of compatibility with previous versions of algorithms. Let's assume we have two algorithms: KZ 1.0 and KZ 2.0; version 1.0 displays five different cell types, while version 2.0 displays only two - and with different marker colors and shapes. KZ 2.0 is vastly better than KZ 1.0, so customers upgrade to version 2.0 and uninstall 1.0 (because these two versions are not allowed to co-exist).

The use-case: what should happen when a 2.0 customer attempts to open the results from a 1.0 study?

While answers might vary, a reasonable one would be "treat the KZ 1.0 results as read-only", i.e. allow the user to display the WSA results as an KZ 1.0 overlay; but how do we know how to visualize KZ 1.0 overlays while the workflow associated with KZ 1.0 has been removed?

DIPLOMAT tries to address the third use-case by including a standardized way to encode how algorithm results should be displayed in a "read-only" mode. The approach is inspired by DICOM's "presentation state", which also attempts to abstract out display parameters but is more limited. DIPLOMAT provides a standard way to define presets ('named' descriptions of marker colors and shapes, color lookup tables, default display settings) and scenes (complex visualizations including multiple layers); this information is internally stored in a self-describing JSON format in a DIPLOMAT group called **wsi_presentation**.

The concept of **wsi_presentation** also enables the final use-case: improved compatibility between clinical platforms. DIPLOMAT results from various Roche platforms can be easily shared; this is read-only display without associated workflow only, but it represents a vast improvement in compatibility.

TEMPLATE: FT0500-504C

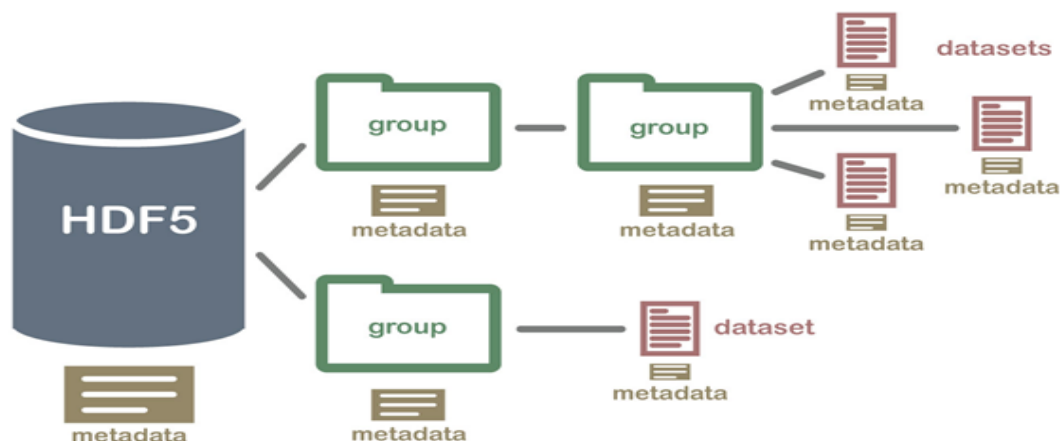
TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

The enabling technologies for read-only display are the JSON-based rendering 'recipes' that are stored in DIPLOMAT and a high-performance dynamic overlay tile generator called OGS (overlay generator service) that is integrated into Roche uPath to enable DIPLOMAT display. OGS is implemented as a docker container and is a service that uses the information in DIPLOMAT files to generate overlay tiles on-the-fly. Roche platforms can integrate OGS in their workflow, which enables users to modify the 'recipes' used by OGS to change display settings (i.e. enable/disable markers, change colors etc.).

OGS will likely be made available in the future to other Roche DP groups; the long-term vision is to move overlay tile generation into the client (browser or a dedicated client). the technology required for this is tentatively called ROSIE (Roche Overlay Server cliEnt). Any algorithm that generates DIPLOMAT output (internal or external to Roche) will be supported by OSG/ROSIE.

Why HDF5?

HDF5 can be thought of as a file system contained and described within one single file; this implies that HDF5 can contain virtually "anything", which makes it a good candidate for a standardized format that needs to adapt to evolving needs.



TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

Figure 1: an example HDF5 file structure which contains groups, datasets and associated metadata ([from the web](#))

Because HDF5 also allows for embedding of metadata, it makes the format *self-describing*. By attaching metadata to every file/group/dataset in the HDF5 file, we do not need a separate (and additional) metadata document; this allows for automation and easy integration in a Python environment.

Roche has successfully used HDF5 in the past to store segmentation masks; this approach relies on the use of “chunking” (slicing) which divides the entire output mask into tiles that are individually compressed and stored. This makes HDF5 a portable, yet efficient file format.

HDF5 files can store “anything; while this makes the format “future-proof”, it also means that we need conventions if we want to use HDF5 as a common exchange format. Describing these conventions behind DIPLOMAT is the main goal of this document.

What about DICOM?

There are good reasons to consider adopting [DICOM](#), which is **THE** standard in medical imaging. Unfortunately, DICOM is a complex standard (the required functionality is split over various DICOM standards for segmentation, presentation state, parametric map images, and others) and is certainly not self-describing. While there DICOM toolkits available that allow DICOM data to be processed in Python, (Geo)JSON-based storage schemes are currently simply much more popular with data scientists.

As there was a need to move fast with respect to DIPLOMAT support, we decided that the initial DIPLOMAT releases (1.x) will not support DICOM – the Roche development team did not have the capacity (or know-how) to implement robust DICOM support within the desired time frame. However, future DICOM support will be considered essential; DIPLOMAT will enable this by providing future support for DICOM representations of binary masks, markers, presentation state and others.

Roche's Overlay Tile Generator Service (OGS)

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

DIPLOMAT is now the primary file format that uPath uses for display of algorithm results. To that end, Roche needed to develop a uPath service that could read DIPLOMAT files and use their contents to generate overlay tiles - this service is internally referred to as OGS. This service converts the contents DIPLOMAT file into a spatial data structure (internally referred to as the spatial data cache) which allows for very fast spatial queries. OGS is using these spatial queries (like "which markers are present in the tile"?) to determine what needs to be drawn. After drawing the required information into a bitmap, OGS converts the generated bitmap into a PNG stream that is then sent to the browser to be overlayed over the pathology image.

OGS is implemented in C++ and has been carefully performance-optimized; it can generate more than a billion overlay pixels per second and the performance bottleneck is not overlay tile generation, but uploading of the generated PNG tiles to the browser.

There is an inherent conflict between the initial concept of DIPLOMAT ("store analysis results in a self-describing format so it can be "anything") and the ability of OGS to actually parse algorithm results and display results. This DIPLOMAT specification will limit examples to those that actually can be correctly parsed by OGS; we anticipate future updates of the OGS DIPLOMAT parser to handle new use-cases while corresponding extensions of the DIPLOMAT specification will also be needed.

Contents of a DIPLOMAT HDF5 file

Algorithms that conform to the Roche Open Environment specification write their algorithm results into a HDF5 file, which can contain several groups:

- **wsi_analysis_info**, which contains information about the algorithm, input data, system info, and other information. **This group is mandatory!**
- **wsi_cells**, which contains results (position, staining, contour, etc.) associated with individual nuclei; this content can be used for spatial queries and overlay generation;
- **wsi_masks**, containing masks that are typically used to detect tumor, stroma, and other areas;
- **wsi_heatmaps**, which contain scalar data that can be used to generate a color overlay that is superimposed over the image;

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

- **wsi_thumbnail**, which contains a static thumbnail that can be used as a navigational tool to locate diagnostic relevant regions of the WSI;
- **wsi_presentation**, which contains the "recipes" on how to convert information from the various groups into an overlay (e.g. default label colors and shapes, annotation colors, convert heatmap scalar data into overlay colors, etc.); this information is encoded in several additional groups under **wsi_presentation**. This group also contains the specification of any panels that control the overlay display.

In Diplomat 1.3, support will be added for

- **wsi_annotations**, which contain annotations generated by the WSA algorithm and/or user annotations that were used as input for the algorithm;
- **wsi_scores**, which contains the whole slide score of the algorithm.

It is allowed to write other groups to a DIPLOMAT file; these are ignored by Roche Open Environment and can be used to store proprietary information.

The **wsa_analysis_info** group is mandatory; it minimally should contain information about the algorithm used, etc. The other groups are optional, although **wsi_cells** and/or **wsi_masks** contents are usually present. The sequel describes the contents of the DIPLOMAT groups in detail.

wsi_analysis_info: Algorithm and runtime information

For now, we will assume that a DIPLOMAT HDF5 file contains the results from exactly one algorithm. The algorithm information should be contained in the group **wsi_analysis_info**; the information should be stored as standard JSON. We recommend capturing as much information about the algorithm and its runtime deployment as possible as it is the obvious place to store information about the runtime performance of the algorithm (proprietary information can be encrypted into JSON or written to a non-DIPLOMAT HDF5 group).

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

In earlier versions of the DIPLOMAT specification, `wsi_analysis_info` consisted of one JSON string that contained all information about the algorithm and its execution. To simplify parser implementation, DIPLOMAT now splits up the JSON into different HDF5 groups:

- **diplomat** (version number, default locale, date/time of file generation, etc., mandatory)
- **developer** (based on Roche Open Environment [Vendor API](#), name and logo of vendor, optional)
- **algorithm** (based on Roche Open Environment [Algorithm Detail API](#), algorithm details; mandatory)
- **input** (information about the WSI input; mandatory)
- **platform** (information about the hardware that executed the algorithm; optional)
- **runtime** (runtime performance metrics; optional)

The diplomat, algorithm and input groups are mandatory, others are optional. If present, a group should contain a UTF-8 encoded JSON string.

`wsi_analysis_info/diplomat`

The three mandatory fields in the diplomat group are the Diplomat version (current value: 1.30), the default locale that was used (default: en-US) and an UUID that uniquely identifies the analysis results (e.g. by using the UUID returned by Start Analysis API, see [Start Analysis](#) of the Roche Open Environment specification). If the diplomat group is not present, the default values are used.

This group can be used to add descriptive information, such as the creation date/time package(s) used to generate the Diplomat file, checksum(s), and other data.

Example of a **diplomat** group:

```
{
```

TEMPLATE: FT0500-504C

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
"version": "1.30",  
  
"locale": "en-US",  
  
"uuid": "27f64d5a-2456-488f-b88b-edea10175c49",  
  
"date": "2022-07-04T13:13:03.969Z"  
  
}
```

wsi_analysis_info/developer

This group provides information about the vendor/developer of the algorithm. It typically includes the company name and logo, but also can include contact information, copyright notices, and other descriptive information relevant to the vendor. This group is optional, but if present, the fields **company_name** and **company_logo** will be used by uPath.

The group name **developer** was chosen in lieu of the for-profit implying "vendor", which is used by Roche Open Environment. **company_name**, **vendor_name**, and **developer_name** are considered equivalent.

Example of a **developer** group:

```
{  
  
"company_name": "Roche",  
  
"company_logo": <embedded jpeg base64 encoded logo>  
  
"address": "Roche Diagnostics, 2881 Scott Blvd, Santa Clara, CA 95050",  
  
"infoURL": "https://diagnostics.roche.com/global/en/products/instruments/upath-her2-dual-ish-image-analysis.html",  
  
"description": "Example Her2 Dual ISH DIPLOMAT file, internal use only"  
  
}
```

wsi_analysis_info/algorithm

This is a mandatory group and provides information about the algorithm. Its contents are based on [Algorithm Detail API](#) of the Roche Open Environment specification., except that **results_parameters**, the JSON object that describes the algorithm parameters for which scores will be generated is not supported

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

by DIPLOMAT 1.3 as there was not sufficient time to do a proof-of-concept implementation (expected to be supported in future) .

Note that several of the algorithm fields define workflow, e.g. whether ROI analysis should be enabled. In DIPLOMAT 1.3, the contents of `result_parameters` can configure the panel that presents the algorithm scores.

Example of an `algorithm` group:

```
{  
  
  "algorithm_description": "Determines ratio of the HER2 gene to Chromosome 17 (Chr17)",  
  
  "algorithm_display_id": "9005",  
  
  "algorithm_id": "9cef739c-8421-4808-ade5-52184ad66b5a",  
  
  "algorithm_name": "HER2 DUAL ISH Breast RUO",  
  
  "algorithm_type": "RUO",  
  
  "clone_type": "CHR17",  
  
  "manual_score_override": false,  
  
  "max_secondary_analysis_allowed": false,  
  
  "overlay_acceptance_required": false,  
  
  "primary_analysis_overlay_display": true,  
  
  "provides_primary_analysis_score": true,  
  
  "provides_prognostic_score": false,  
  
  "requires_analysis_rejection_feedback": false,  
  
  "roi_analysis_support": true,  
  
  "secondary_analysis_support": false,  
  
  "slide_score_acceptance_required": false,  
  
  "software_build": "8.0.60.201",  
  
  "supported_image_formats": [  
  
    "BIF",  
  ]  
}
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
        "DCM",  
  
        "TIF"  
  
    ],  
  
    "supported_magnification": 40,  
  
    "supported_scanners": [  
  
        "VENTANA DP 200",  
  
        "VENTANA DP 600"  
  
    ],  
  
    "tissue_type": "Breast",  
  
    "vendor": "Roche",  
  
    "version_number": "2.0"  
  
}
```

In the above example, undefined fields (`indicationType`, `required_slide_types`, `supported_mpp_ranges` amongst others) were omitted. The fields displayed in red need further elaboration:

- **algorithm_display_id**: the “readable” version of the **algorithm_id**, which is a unique UUID, but is impossible to remember. This ID will be used to identify the dictionary that should be used for translation (to be described in the sequel).
- **software_build**: unlike the **version_number** of the algorithm, software build numbers often change daily during software development. This field is an unique identifier that can help the developer to identify which software version created the Diplomat file.
- **vendor**: this field is redundant (already provided as part of the **developer** entries), but supported as Roche Open Environment 1.3 prescribes it.

While the **algorithm** group is mandatory, not all fields are mandatory or even required. For example, the **supported_image_formats** field is completely

TEMPLATE: FT0500-504C

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

unnecessary as the Diplomat file contains the results of the image analysis, so whatever the input format was, the algorithm accepted it. However, many fields are used to define the workflow that uPath should follow to control display and secondary analysis of the WSI data; especially this aspect of Diplomat needs to be further defined and fleshed out in subsequent versions. As a guideline for Diplomat 1.3, please consider any JSON field that is a simple string to be mandatory.

wsi_analysis_info/input

This is a mandatory group that contains information about the input dataset that was processed. Here is an example:

```
{  
  
  "image_location": "c:/TestData/onedualish/data/N2_DISH_0050.bif",  
  
  "image_id": "0",  
  
  "sha256": "e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855",  
  
  "scanner_name": "VENTANA DP 200",  
  
  "scanner_unit_number": "",  
  
  "microns_per_pixel_x": 0.25,  
  
  "microns_per_pixel_y": 0.25,  
  
  "slide_magnification": 40,  
  
  "slide_width": 40752,  
  
  "slide_height": 38912,  
  
  "slide_depth": 0,  
  
  "number_levels": 9,  
  
  "dimensions": [ [40752,38912], [20376,19456],[10188,9728],  
                  [5094,4864],[2547,2432],[1274,1216],[637,608],[319,304],[160,152]]  
}
```

As a guideline, consider any field that has a numeric value (including dimensions) as mandatory. We need the WSI dimensions to be able to generate a correctly-sized overlay; this data is readily available when the algorithm is executed (as the input WSI needs to be encoded), but will be hard or

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

difficult to obtain if not embedded in Diplomat. We therefore require embedding of the dimensions of the resolution pyramid, the (redundant) slide width and slide height, and the microns per pixel for both x and y (otherwise quantitative measurements that depend on pixel size are not possible). **Note that the DIPLOMAT convention for any coordinate is (x,y); the resolution pyramid dimensions should be in that order, as well as any other coordinates.** The input field needs to contain at least one entry that makes it possible to uniquely identify the input WSI that was used; however, ***no patient-traceable information is allowed to be included.***

ws_i_analysis_info/platform

This is an optional group that is used to specify the hardware used to run the algorithm. This group is for informational purposes only and can be very useful when debugging issues during deployment. It is up to the algorithm developer to decide whether to populate this group and which fields should be used, for example:

```
{  
  
  "OS": "Ubuntu Linux",  
  
  "version": "20.04",  
  
  "device_name": "Z8-342G32",  
  
  "CPU": "Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz (2 processors)",  
  
  "RAM": "96.0 GB (95.7 usable)",  
  
  "device_id": "8774C42-7320-4D86-B215-35EFC596CEE",  
  
  "product_id": "00391-70000-00000-AA735",  
  
  "manufacturer": "Hewlett-Packard",  
  
  "GPU": "NVIDIA RTX A6000"  
}
```

ws_i_analysis_info/runtime

This is an optional group that can be used to capture performance metrics, for example:

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
{  
  
  "uuid": "d95c57d0-d48a-11eb-b8bc-0242ac130003",  
  
  "start_UTC": "2021-06-05T13:15:30Z",  
  
  "end_UTC": "2021-06-05T13:33:20Z",  
  
  "execution_seconds": 1044.23,  
  
  "performance_metrics":  
  {  
  
    "File_load": 32.3,  
  
    "filesize": 1027.3,  
  
    "color_profile": 56.56,  
  
    "init": 2.53,  
  
    "tiling": 42.3,  
  
    "processing": 831.3,  
  
    "hdf5_output": 75.23  
  
  },  
  
  "output_file": "13.21.66.223:/results/ER/T0000313377_GenBreast_IntRepro_PDL1_DP1_Scan3.h5"  
  
}
```

The sole purpose of this group is to enable developers to capture performance metrics for subsequent analysis. Its contents need to be valid JSON, but can be proprietary (e.g. by encrypting performance metrics and then encoding it as a JSON string).

wsi_cells: Storage of spatial data (geometry)

The output of many WSI algorithms is a list of cell positions with corresponding cell types; this information should be easily accessible from a Python environment and therefore encoded in a self-describing, standard format.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

DIPLOMAT uses [GeoJSON](#) to store geometry data; it is an IETF standard for which [validators](#) are available online, it is supported by standard tools like PostGIS and Shapely, and is also a preferred way to [export data from the public domain package Qupath](#). For a good description, see [here](#). Because we are using GeoJSON for storage of spatial data, we can use well-known tools to enable spatial queries (like cell counts, cell area, etc.). There are many utilities available to manipulate and use GeoJSON, see for example the [awesome geojson](#) repository.

How to efficiently store GeoJSON data

GeoJSON is a very verbose format; storing results from the entire WSI in one large GeoJSON string is inefficient both in processing time and storage space. During development of the DIPLOMAT format, we tried several approaches to storing GeoJSON data and we eventually settled on the following approach:

1. As the WSI typically is processed tile-wise, also store the generated GeoJSON information tile-wise.
2. Store the GeoJSON string in a compressed format.

Tile-based GeoJSON

As WSA algorithms usually divide the input into tiles that are processed individually, it should be easy to store tile results in individual GeoJSON files. This has two advantages:

- Potential "on-demand" loading of tile results
- Parallel processing of tile results

To avoid having to parse all tile GeoJSONs to find bounding boxes associated with the results, DIPLOMAT stores an array of bounding boxes associated with the various tile GeoJSONs in a separate dataset named `wsi_cells/index`. An on-demand loader of tile results can use the `index` dataset to determine which JSON files need to be decoded; bounding boxes are **not allowed to have overlap** as that may lead to duplicate results from border cells that are included multiple times. The bounding box coordinates are specified as `[Xleft,Ytop,Xright,Ybottom]`, `Xright` and `yBottom` are included in the bounding box.

Here is an example of an index dataset:

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
[{"filename":"tile2_15","bbox":[2048,15360,3071,16383]},{ "filename":"tile3_15","bbox":[3072,15360,4095,16383]},{ "filename":"tile3_14","bbox":[3072,14336,4095,15359]},{ "filename":"tile4_12","bbox":[4096,12288,5119,13311]},{ "filename":"tile5_13","bbox":[5120,13312,6143,14335]},{ "filename":"tile4_13","bbox":[4096,13312,5119,14335]}]
```

If tiles do not generate any output results, they don't need to be stored either and therefore can be missing from the index. It is up to the developer to choose the position and size of the bounding boxes; likewise, the developer is responsible for choosing the filenames used to store the tile contents.

There is another advantage to tile-based storage; multiple threads can be used for parsing of the JSON streams, which can make loading of the Diplomat file very fast.

Compressed GeoJSON

GeoJSON is very verbose, but can be easily compressed using gzip. Unfortunately, if we store the gzip'd stream as binary data in HDF5, it is the responsibility of the reader client to decode the binary stream, which then would deviate from the goal of making the format self-describing.

Fortunately, HDF5 can do the encoding and decoding for us; to make that happen, we need to store the JSON string as "chunked" with the chunksize being equal to the length of the string. Here is the General Object Info (using HDFView) of a tile:

Dataset Dataspace and Datatype	
No. of Dimension(s):	1
Dimension Size(s):	1
Max Dimension Size(s):	1
Data Type:	String, length = 200862, padding = H5T_STR_NULLTERM, cset = H5T_CSET_ASCII
Show Data with Options	
Miscellaneous Dataset Information	
Storage Layout:	CHUNKED: 1
Compression:	4.543:1GZIP: level = 6
Filters:	GZIP
Storage:	SIZE: 44214, allocation time: Incremental
Fill value:	NONE

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

By applying chunking, the GeoJSON is reduced from 200862 bytes to 44214 bytes, which is a 4.543x compression.

Example: list of cell positions and associated labels

Many WSI algorithms produce a list of nuclei positions and associated labels. How would we encode this in GeoJSON? In fact, this is pretty straightforward. As the number of labels is typically small, one should organize the GeoJSON to list the positions per label as follows:

```
{ "name": "Ki67_RU0", "type": "FeatureCollection", "features": [ { "type": "Feature", "properties": { "label": 0 }, "geometry": { "type": "MultiPoint", "coordinates": [ [ 12466, 9445 ], [ 13048, 9253 ], [ 13086, 9279 ], [ 12552, 9601 ], [ 12747, 9794 ], [ 12603, 9930 ], [ 12883, 9796 ], [ 12437, 10093 ], [ 12473, 10098 ], [ 12474, 10108 ], [ 12434, 10109 ], [ 12531, 10219 ], [ 12740, 10207 ] ] } }, { "type": "Feature", "properties": { "label": 1 }, "geometry": { "type": "MultiPoint", "coordinates": [ [ 12962, 9217 ], [ 12985, 9218 ], [ 12920, 9221 ], [ 12946, 9223 ], [ 12898, 9238 ], [ 13004, 9244 ], [ 12981, 9275 ], [ 12971, 9392 ], [ 13005, 9425 ], [ 13009, 9436 ], [ 12960, 9469 ], [ 13126, 9257 ], [ 13063, 9259 ], [ 13077, 9262 ], [ 13087, 9262 ], [ 13109, 9275 ], [ 12652, 9580 ], [ 12641, 9594 ], [ 12586, 9608 ], [ 12743, 9611 ], [ 12772, 9611 ], [ 12629, 9612 ], [ 12565, 9618 ], [ 12544, 9621 ], [ 12757, 10097 ], [ 12767, 10212 ] ] } }, { "type": "Feature", "properties": { "label": 2 }, "geometry": { "type": "MultiPoint", "coordinates": [ [ 12517, 9229 ], [ 12537, 9253 ], [ 12381, 9277 ], [ 12530, 9295 ], [ 12501, 9329 ], [ 12534, 9335 ], [ 12322, 9353 ], [ 12509, 9353 ], [ 12408, 9356 ], [ 12297, 9369 ], [ 12515, 9372 ], [ 12387, 9374 ], [ 12497, 9379 ], [ 12493, 9385 ], [ 12407, 9386 ], [ 12309, 9388 ], [ 12349, 9388 ], [ 12514, 9402 ], [ 12317, 9405 ], [ 12433, 9406 ], [ 12452, 9414 ], [ 12322, 9417 ], [ 12404, 9419 ], [ 12493, 9419 ], [ 12479, 9421 ] ] } } ] }
```

For each cell label, we need to define a GeoJSON feature with associated property "label" and then provide MultiPoint geometry that contains the coordinates of the corresponding cells. Note that this representation is space-efficient, especially when chunking is used; chunking will compress the (condensed) example above by a factor of approximately 3x.

Example: list of cell positions, associated labels, and stain intensities

In Roche's PR algorithm, the algorithm produces a list of nuclei positions, associated labels, and, in case the nucleus is tumor+, a staining value. How would we encode this in GeoJSON?

Unfortunately, the answer is that this can be done in several ways. DIPLOMAT therefore uses conventions:

1. Separate the nuclei by label values: aside from the label value, the only difference between nuclei is their position; in that case, use a MultiPoint to list positions.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

2. If nuclei have associated information (like a stain value or area), each nucleus must be a separate feature containing exactly one coordinate. In case of ER, we would store the label value and stain value for each nucleus.
3. Make the name/value of the properties short and descriptive; this minimizes the raw size of the GeoJSON. DIPLOMAT uses presentation state to convert names and values to more meaningful names (like a label value of 0 meaning "Tumor Positive"). Forcing this indirection will help to facilitate internationalization/translation of algorithm results.

Here is an example of an encoding where label 0 cells (TC+) need to contain additional information about staining:

```
{"name": "PR_RUO", "type": "FeatureCollection", "features": [{"type": "Feature", "properties": {"label": 0, "stain": 73}, "geometry": {"type": "Point", "coordinates": [4144, 8229]}}, {"type": "Feature", "properties": {"label": 0, "stain": 95}, "geometry": {"type": "Point", "coordinates": [4315, 8259]}}, {"type": "Feature", "properties": {"label": 0, "stain": 57}, "geometry": {"type": "Point", "coordinates": [4326, 8295]}}, {"type": "Feature", "properties": {"label": 0, "stain": 94}, "geometry": {"type": "Point", "coordinates": [4313, 8314]}}, {"type": "Feature", "properties": {"label": 1, "geometry": {"type": "MultiPoint", "coordinates": [[4124, 8207], [4097, 8229], [4116, 8251], [4104, 8262], [4277, 8293], [4305, 8329], [4187, 8336], [4260, 8355], [4187, 8361], [4201, 8366], [4197, 8379], [4321, 8386], [4219, 8391], [4337, 8424], [4279, 8429], [4198, 8433]]}}, {"type": "Feature", "properties": {"label": 2, "geometry": {"type": "MultiPoint", "coordinates": [[4157, 8194], [4291, 8194], [4337, 8201], [4197, 8205], [4157, 8209], [4336, 8210], [4216, 8212], [4173, 8213], [4240, 8213], [4304, 8217], [4188, 8219], [4219, 8228], [4295, 8229], [4212, 8241], [4200, 8247], [4336, 8249], [4187, 8257], [4325, 8257], [4205, 8259]]}}}]
```

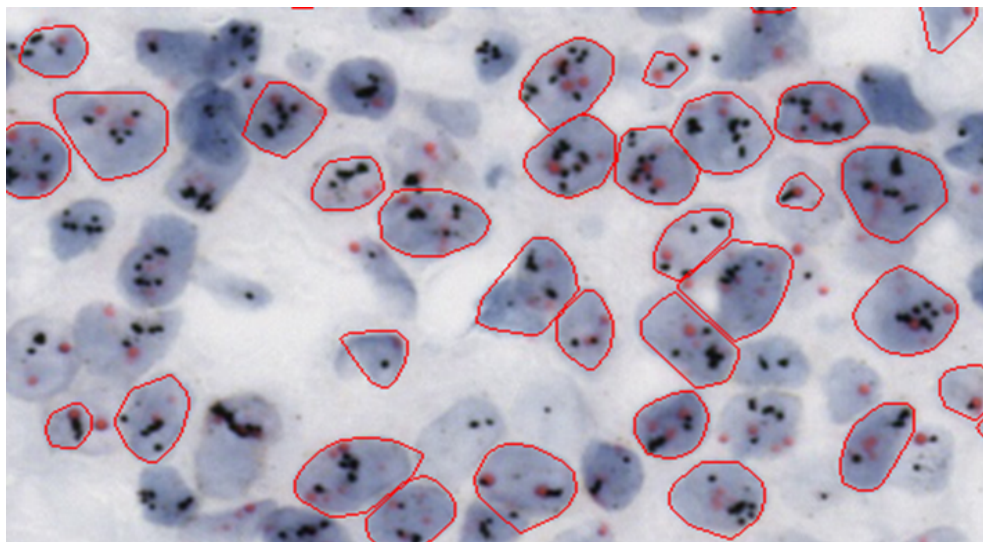
Note that we use the properties field to describe the type of cells and also the stain intensity of each tumor+ cell. While the GeoJSON file above is valid, properties have no meaning to GeoJSON (it is a "Foreign Member" according to the spec); we need to specify which properties we would like DIPLOMAT to support. The name of this specification is "PR_RUO", which will be referenced by the `ws_i_presentation` entry.

Note that the specification of the label 0 cells is very verbose, with many strings that need to be repeated for each cell. This is best-case for gzip compression, which excels at finding these repeating patterns; for this tile, the JSON was compressed from 64444 to 8808 bytes (7.3x compression).

Several browser plugins support GeoJSON and provide hooks that allow the drawing of markers to depend on a property value (like "label", in this example). The recipe on how to do that should be part of the presentation state description in the DOCUMENT HDF5 file.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

Example: Roche Dualish (cell boundaries, black and red dot count)



For the DualISH algorithm, we might store cell boundary contours, the center position of the cell, and the number of black and red dots for each cell. This is an example of how DualISH stores the algorithm results:

```
{
  "name": "DualISH_RU0",
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "label": 1,
        "geometry": {
          "type": "MultiPoint",
          "coordinates": [
            [11448, 17404], [11512, 16410], [11509, 16414], [11509, 16417], [11639, 17334], [11640, 17336], [11643, 17339], [12082, 16639], [11658, 16883], [12132, 16991]]
          ]
        },
      "type": "Feature",
      "properties": {
        "label": 2,
        "geometry": {
          "type": "MultiPoint",
          "coordinates": [
            [11354, 16387], [11623, 17389], [11386, 17393], [11555, 17399]]
          ]
        },
      "type": "Feature",
      "properties": {
        "blk": 2,
        "red": 1,
        "score": "1",
        "geometry": {
          "type": "Polygon",
          "coordinates": [
            [[12013, 17062], [12013, 17067], [12012, 17068], [12012, 17069], [12010, 17071], [12010, 17072], [12007, 17075], [12007, 17076], [11996, 17087], [11995, 17087], [11987, 17095], [11986, 17095], [11985, 17096], [11983, 17096], [11977, 17087], [11994, 17058], [11996, 17058], [11997, 17057], [12008, 17057], [12013, 17062]]
          ]
        },
      "type": "Feature",
      "properties": {
        "blk": 2,
        "red": 1,
        "score": "0.97985846",
        "geometry": {
          "type": "Polygon",
          "coordinates": [
            [[11753, 17236], [11753, 17238], [11752, 17239], [11752, 17241], [11751, 17242], [11735, 17237], [11736, 17236], [11737, 17236], [11738, 17235], [11752, 17235], [11753, 17236]]
          ]
        },
      "type": "Feature",
      "properties": {
        "blk": 2,
        "red": 1,
        "score": "0.95413715",
        "geometry": {
          "type": "Polygon",
          "coordinates": [
            [[12144, 17026], [12144, 17027], [12145, 17028], [12145, 17029], [12146, 17030], [12146, 17031], [12147, 17032], [12147, 17033], [12136, 17027], [12137, 17027], [12139, 17025], [12143, 17025], [12144, 17026]]
          ]
        },
      "type": "Feature",
      "properties": {
        "blk": 3,
        "red": 1,
        "score": "0.37765378",
        "geometry": {
          "type": "Polygon",
          "coordinates": [
            [[11760, 17139], [11760, 17140], [11761, 17141], [11761, 17142], [11762, 17143], [11762, 17145], [11763, 17146], [11763, 17148], [11764, 17149], [11764, 17150], [11765, 17151], [11765, 17153], [11759, 17159], [11758, 17159], [11755, 17162], [11754, 17162], [11752, 17164], [11751, 17164], [11741, 17140], [11743, 17140], [11744, 17139], [11746, 17139], [11747, 17138], [11753, 17138], [11754, 17137], [11758, 17137], [11760, 17139]]
          ]
        }
      }
    ]
  }
}
```

The JSON code in red identifies coordinates of features with label 1 and label 2; the remaining code stores the analysis result for each cell, which consists of a count of blk (black) and red dots, as well as a score.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

Note that the Polygon specification of the cell contour should always be specified in a counter-clockwise fashion. The name of this specification is "DualISH_RUO", which will be also referred to by **wsi_presentation**.

This concludes the examples of GeoJSON-based storage of cell results. In Diplomat 2.0, we plan to add support DICOM supplement 222 (Microscopy Bulk Simple Annotations Storage SOP class); this supplement supports a collection of points, closed polygons, open polylines, and simple geometric shapes.

wsi_masks: Storage of segmentation masks

For segmentation masks, we take advantage of HDF5 support for compressed "chunked" data. HDF5 can internally tile datasets; conceptually, it uses the file system as backing storage and will compress and write individual tiles as needed. The default compression algorithm used by "chunking" the image is gzip, which works very well on artificial data like masks that do not contain noise.

HDF5-based compression has been successfully used for storage of WSI masks by Roche for several years and DIPLOMAT adopts that approach.

Binary masks

Segmentation masks are simple 2D arrays of numerical labels, of size **uint8**, where each entry represents a pixel on a specific slide pyramid level. This allows for a range of 255 possible labels, with the value of 0 always being considered background as a convention.

Any algorithm that wishes to output masks has to containerize all output masks in a group using the key **wsi_masks**. DIPLOMAT support three flavors of segmentation masks:

- Single label masks, where each pixel in the mask can have only one label and a maximum of 254 labels is supported.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

- Multi-label masks, where each pixel in the mask can have more than one label. This is useful especially for detecting segmentation labels that may overlap, thus yielding regions that can have two or more labels associated with them. Each label in a multi-label mask is stored in a separate dataset.
- Hybrid multi-label masks, where each pixel in the mask can have more than one label, but labels are stored using bit planes of one dataset.

Single label segmentation masks

All mask members of the `wsi_masks` group for single label masks should be saved as dataset objects following the naming convention below:

- `wsi_masks/predicted_region_mask_l{pyramid_level}`: a mask at the specified pyramid level, where 0 is the native image resolution.
- `wsi_presentation/masks` should contain a JSON description that includes the mask name and its color.

In the various examples, `predicted_region_mask` is the mask name; we could use other identifiers, like `wsi_mask/istissue_l3` which would contain a mask indicating tissue areas at resolution level 3.

An example: We would like to create a mask which detects two mutually exclusive segmentation areas: tumor nest area and stromal area. The mask should contain the following keys:

- `wsi_masks/predicted_region_mask_l0`, where every stromal area pixel will be set to the value 1, every tumor nest area pixel will be set to 2, and every other pixel to 0
- `wsi_presentation/masks` contains a JSON description that not only contains name and color of the tumor and stromal areas, but also other fields which will be explained later:

```
{  
  
  "default": [{  
  
    "maskname": "predicted_region_mask",
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
        "textgui": "stromal",

        "label": 1,

        "locked": false,

        "visible": true,

        "color": "rgba(0, 255, 0, 255) ",

        "level_opacity": [0.15, 0.2, 0.25, 0.3, 0.35, 0.4],

        "level": -1

    },

    {

        "maskname": "predicted_region_mask",

        "textgui": "stromal",

        "label": 2,

        "locked": false,

        "visible": true,

        "color": "rgba(0, 255, 0, 255) ",

        "level_opacity": [0.15, 0.2, 0.25, 0.3, 0.35, 0.4],

        "level": -1

    }

]

}
```

The above JSON code defines the preset **default** which describes how the tumor and stromal mask should be presented; it also provides hooks that can be used to allow an application GUI to build a menu with a preset name and a description of the preset (see the description of **ws_i_presentation** for more information). In this case, the presentation JSON specifies that the user is allowed to edit the mask color assignments ("locked" is false), the mask is visible by default ("visible" is true) and the masks should be rendered for each resolution level ("level" is -1, which means "all levels"; we also could have provided a list of the resolution levels that should display a mask).

TEMPLATE: FT0500-504C

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

Finally, the `level_opacity` list contains an opacity multiplier associated with each pyramid level; at resolution level 0 (original resolution), the mask opacity will be multiplied by 0.15, which yields a very transparent mask. At higher levels of the resolution pyramid, the mask opacity will increase.

The preset specification can be omitted (i.e. only an array of the mask specifications is present).

An important note: **only provide masks at one resolution level** (masks can be efficiently downsampled and upsampled, so only store masks at the highest resolution level used for mask display).

Multi-label segmentation masks

Multi-label masks use the following naming convention for storing data:

- `wsi_masks/predicted_region_mask_l{pyramid_level}_{label}`: region mask at the specified pyramid level for the label specified.
- `wsi_presentation/masks` contains the name and color associated with the region mask.

An example: we would like to create a mask which stores two segmentation areas that are not mutually exclusive: pen marker and out of focus areas. We therefore need two separate masks:

- `wsi_masks/predicted_region_mask_l0_1`: mask for pen markers (set to 1)
- `wsi_masks/predicted_region_mask_l0_2`: mask for out of focus (set to 2)
- `wsi_presentation/masks` contains the names and colors associated with the two masks (similar to the previous example).

Hybrid multi-label segmentation masks

Many algorithms only generate a few masks that might overlap; we can store the bits of those as separate masks or combine all bits into a bit-plane mask, which is called a hybrid segmentation mask. A hybrid mask of type `int8` (the default) can store a maximum of 8 multi-label masks, while other `uint`

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

types can support more than 8 bits. Hybrid masks use the following naming convention:

- `wsi_masks/predicted_region_mask_hybrid_l{pyramid_level}_{startlabel}`: store up to eight region masks starting with label `startlabel`. The start label should be 0 or a multiple of eight, so `predicted_region_mask_hybrid_l1_8` would store level 1 region masks 8 through 15.

It is possible to store less than eight masks in a hybrid mask; the unused bitplanes should be set to 0. Color and name specifications in `wsi_presentation/masks` are done identically compared to non-hybrid masks.

The main advantage of hybrid masks is their 8-bit values can index into a 256-entry color lookup table, which can greatly speed-up display as it maps very well on GPU hardware (i.e., 1D texture lookup in a fragment shader).

This concludes the description on how to store masks in Diplomat. In Diplomat 2.0, we plan to add support for DICOM rasterized segmentations as described in DICOM supplement 111.

wsi_heatmaps: Heatmaps

"Heatmaps" are intended to be overlaid on top of the WSI image, usually with some transparency. The initial support of heatmaps was modelled after the approach currently used by Roche, but Diplomat now also supports [SVG](#) and an additional Roche proprietary format that will be made public in a future Diplomat version.

Heatmap overlays stored as SVG

SVG is an abbreviation of "Scalable Vector Graphics"; it is an XML-based format that stores geometrical primitives rather than images. In Diplomat, the SVG image is assumed to be in the original WSI image resolution; scaling SVG to lower resolutions is fast without loss in quality.

SVG heatmaps are stored in the HDF5 group `/wsi_heatmaps/<name>.svg`; for example, the Diplomat file can contain `wsi_heatmaps/Cancer.svg` and

TEMPLATE: FT0500-504C

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

`wsi_heatmaps/Inflammation.svg`. As SVG files are basically XML, they contain many repeated fragments of text, so they are well suited for gzip-based compression. The various binary svg streams should therefore be individually encoded as an UINT8 dataset with a chunk size equal to the size of each SVG stream.

Roche's OSG overlay tile generator is using the LunaSVG C++ library to render the SVG information; any XML metadata embedded in the SVG will be ignored. Just like with mask images, information on how to render the overlay should be present in the `wsi_presentation` group. An appropriate `wsi_presentation/heatmaps` for the svg-based heatmap example would be:

```
[
  {
    "textgui": "Cancer",
    "name": "Cancer.svg",
    "colorlegend": "jet",
    "visible": true,
    "locked": false,
    "level_opacity": [ 0.15, 0.2, 0.25, 0.3, 0.35, 0.4 ]
  },
  {
    "textgui": "Inflammation",
    "name": "InflammationAll.svg",
    "colorlegend": "jet",
    "visible": false,
    "locked": false,
    "level_opacity": [ 0.15, 0.2, 0.25, 0.3, 0.35, 0.4 ]
  }
]
```

The `textgui` entry is a key to a dictionary that supports locale-specific translations. The SVG was generated using a color scale named "jet", which

TEMPLATE: FT0500-504C

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

should be present in `wsi_presentation/colorluts`; `colorlegend` indicates that the jet scale should be provided as a legend in the display (if the color lookup table needs to be applied, a `colorlut` entry would have been specified). For SVG heatmaps, level-specific opacities are supported as well.

Heatmaps stored as chunked HDF5 tiles

The simplest format of a heatmap is a simple 2D array of numerical values, of any size up to `float32`, where each entry represents a pixel on a specific slide pyramid level. Just as with binary masks, the heatmap should be written out as HDF5 "chunks" (in this case tiles), usually with (gzip) compression enabled. They are stored in the HDF5 group `/wsi_heatmaps` as follows:

- `/wsi_heatmaps/heatmap_{number, starting from 0}/heatmap_l{pyramid level}`, which contains a 2D array of values;
- `/wsi_presentation/heatmaps` should contain further information on how to display the heatmap (including a color lookup table).

The HDF5 group `wsi_presentation/heatmaps` will be described later.

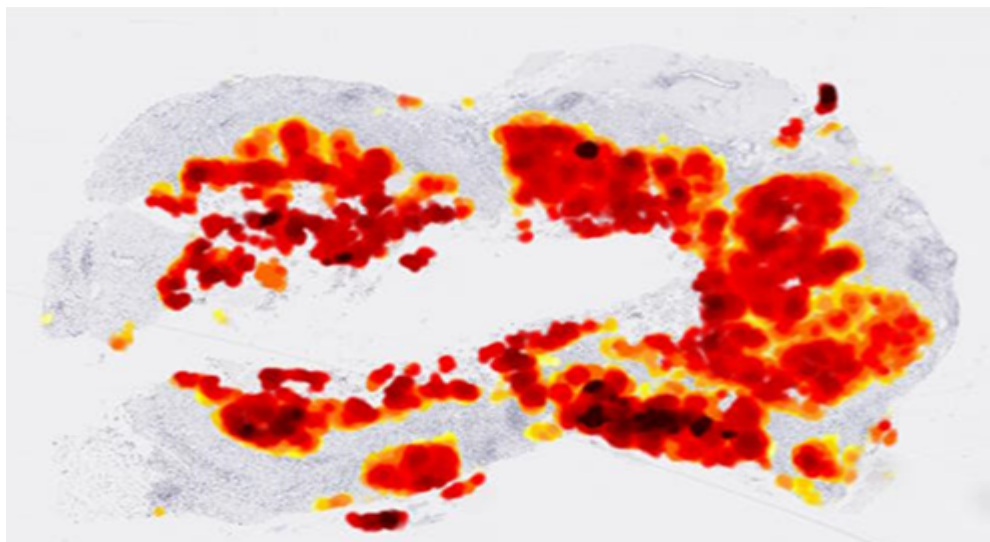
Heatmaps stored as DICOM

There is not a DICOM IOD (Information Object Definition) available for heatmaps, but if a heatmap is considered a probabilistic segmentation, we can use DICOM rasterized segmentations as described in DICOM supplement 111 for storing heatmaps. We will explore potential support in DIPLOMAT 2.0.

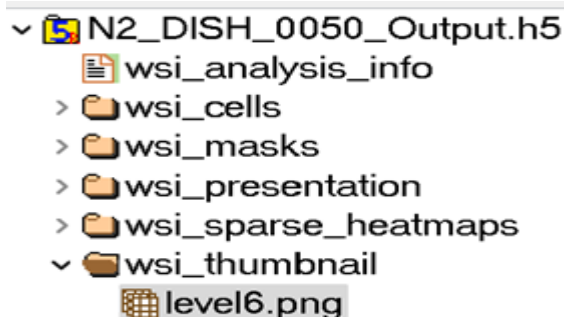
wsi_thumbnail: Thumbnail

A thumbnail is a low-resolution image that is used for navigational purposes - but it is simply a lower-resolution version of the WSI (in png format). We can overlay a heatmap over the thumbnail, which then shows potentially interesting areas of interest in the WSI. Such thumbnail heatmaps are useful for algorithms like Roche's DUALish:

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3



A thumbnail should be large enough to distinguish interesting areas of the WSA; we recommend a size of approximately 1K pixels for the largest dimension. The thumbnail is usually calculated by extracting low-resolution tiles from the WSI resolution pyramid and then stored as a PNG.



In the example above, we store a thumbnail from pyramid level 6 of the dataset as a binary 8-bit integer stream.

Upon the initial load of the dataset, OSG will load the thumbnail in memory and then, if present, overlay the default heatmap(s) at the appropriate resolution level and re-encode the generated thumbnail heatmap as a PNG. This approach enables configurable thumbnail heatmaps.

wsi_presentation: Presentation State

DIPLOMAT attempts to decouple storage of algorithm results from the way that they should be displayed. This approach was inspired by the DICOM standard
TEMPLATE: FT0500-504C

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

for Presentation State, which contains information on how a particular image should be displayed. The `ws_i_presentation` group contains the "recipes" of how the WSA results should be displayed; this includes overlay colors, lookup tables, and marker sizes and shapes. Vendors can use presentation state to encode the preferred way that information should be displayed; `ws_i_presentation` also provides the necessary hooks for internationalization of the presentation of the algorithm results.

Clinical systems should be able to override the presentation state encoded in the algorithm results. For example, users might want to change the label colors associated with individual nuclei or even hide specific marker types; this is equivalent to a change/update of the presentation state. Note that DIPLOMAT files should be treated as read-only after the algorithm generated them; any user-driven changes (hiding markers, modifying colors, editing cell labels, etc.) should be stored separately from the DIPLOMAT files (usually in a database).

The concept of presentation state basically assumes the ability of generating tiles dynamically; for example, while there are various ways to implement enabling/disabling the display of specific marker types (multiple static overlays, simple color LUT operations, etc.), it is not possible to change marker sizes and shapes without dynamic tile generation. Geospatial applications like Google Maps or Openstreetmap typically rely on browser functionality to generate tiles dynamically (using WebGL or similar technologies); one can also generate and encode tiles on servers which are then downloaded and displayed on the viewer. Roche OGS (Overlay Generation Service) enables DIPLOMAT display in Roche uPath.

As `ws_i_presentation` attempts to decouple actual results data from how it is presented to the user, it also is used to implement internationalization (i.e. translations into different languages), which is described next.

Support for internationalization: `ws_i_presentation/locales/...`

The enabler for internationalization is "don't code strings directly in code, use indirection". This is exactly the approach that DIPLOMAT takes; all strings that might end up in a GUI need to be clearly identified, while a

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

mapping into the language in the algorithm also needs to be provided in the DIPLOMAT file.

Roche algorithms typically are developed in English-speaking countries; there is a mandatory entry in `ws_i_analysis_info/algorithm/locale` that needs to specify the algorithm locale, in Roche's case "en" or "en-US". Together with other information from `ws_i_analysis_info`, we can determine the name of a dictionary that should be used for translating internal strings to GUI-displayed strings; the dictionary name is `{vendor}_{display_algo}_{version}_{locale}` (any whitespace character in that name needs to be replaced by an underscore). Furthermore, DIPLOMAT mandates that the algorithm provides a dictionary in its default language in `ws_i_presentation/locales/<locale>`; so, for example, Roche's DUALish RUO algorithm would provide a dictionary in `ws_i_presentation/locales/en-US/Roche_9005_1.1.0_en-US`.

The dictionary contains a mapping from internally used strings (acting like 'keys') to strings to be displayed on the GUI (the 'values'). Any entry in any GeoJSON or JSON group specified as a "textgui" or "key" will be marked as a key that has an entry in the dictionary. The dictionary should contain the string that should be used in the GUI, the units that should be displayed and a more extensive description of the meaning of the field.

uPath currently supports 21 different languages and may support more locales in the future; the supported locales as well as the fields to be translated as per locale can be found in the Roche Open Environment specification [section 8](#).

An example undoubtedly would help. For each nucleus, Roche's DUALish algorithm associates the following properties with each nucleus:

```
{ "blk": "3", "red": "1" }
```

We would like to provide enough hooks in DIPLOMAT to allow the GUI to show the numbers and meaning of blk and red when the cursor would hover over a

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

nucleus. To that end, we need to indicate (as part of a JSON file in `wsi_presentation`, description to follow) that these field names need to be remapped/translated:

```
"textgui": [ "blk", "red" ]
```

The `wsi_presentation/locales/en-US/Roche_9005_1.0.0_en-US` dictionary could contain the following entries:

```
{
  "blk":
  {
    "gui": "HER2",
    "units": "",
    "description": "Number of HER2 probes detected"
  },
  "red":
  {
    "gui": "CHR17",
    "units": "",
    "description": "Number of Chromosome 17 probes detected"
  }
}
```

In case a deployment platform uses a different locale (e.g. `"de_DE"`), we would need a different dictionary (`Roche_9005_1.0.0_de_DE.json`) - which is not part of the DIPLOMAT file. In that case, the dictionary needs to be provided by another means (e.g., by using software hooks that auto-download a dictionary on demand from a Roche website, or a dictionary that get installed

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

when an algorithm gets installed in uPath); those implementation details will vary and are outside the scope of DIPLOMAT.

The key advantage of the proposed approach is that internationalization is "built-in" into DIPLOMAT; adding support for additional languages does not require revalidation/testing of algorithms, but only providing a locale-specific dictionary (which can be automated).

Color specification

DIPLOMAT decouples the storage of scalar values in heatmaps from the color assignments in the overlay; which is assumed to be done using a color lookup table (LUT for short). As we'd like to potentially take advantage of graphics hardware if present, DIPLOMAT can describe the LUT in a format that is usually used by GPUs (eg., by OpenGL):

1. RGB(A) lookup tables are usually stored as textures that are addressed with texture coordinates that have a range between 0.0 and 1.0;
2. Color values are also between 0.0 and 1.0.

RGB colors in JSON are often represented in a hexadecimal format, e.g. "#00FF00" represents a pure green color. Unfortunately, adding transparency as a fourth component is not supported in GeoJSON, "#00FF00FF" can yield a GeoJSON error - avoid hexadecimal formats for RGBA colors.

1. When using integer numbers, the range of color values is between 0 and 255. A pure red RGB color can be specified by "rgb(255,0,0)";
2. When using floating point numbers, the color range is between 0.0 and 1.0. Pure red can also be specified by "rgb(1.0,0.0,0.0)";
3. When adding transparency, rgba tuples should be used, e.g. "rgba(1.0,0,0,0.5)" specifies partially transparent red. Note that [rgba\(255,0,0,0.5\)](#) is identical.
4. A list of color components is also supported; "[255,0,0,0.5]" is also a valid RGBA color specification.
5. A final way to specify colors is using a string. Many colors have [standardized names](#), so we also can specify "red,50%" to specify a partially transparent pure red color.

Color LUT specifications: wsi_presentation/colorluts

As a convention, JSON specification of color LUTs that use integer variables as input (usually 8 and 16-bit) should use integer bins, while floating point input (usually float32) should use floating point bins. For example, a

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

`wsi_mask` that stores label values is stored as an 8-bit integer mask and therefore we should specify the integer numbers when mapping them on specific colors:

```
{  
    "1": "rgb(255,255,0) ",  
    "2": "rgb(0,255,0)",  
    "3": "rgb(0,255,255)"  
}
```

From a GPU point of view, the actual LUT would use the texture coordinate associated with the entry of 2 ($2/255 = 0.00784314$) to map onto a white color - but having to specify the floating point texture coordinate would be too unwieldy and therefore the use of integers is supported for masks.

It is also important to specify how colors need to be interpolated. For example, let's assume that we display an overlay of a labelled mask where a bin value of 1 corresponds to red and a bin value of 2 corresponds to green. With a high magnification, we might end up with mask values that are bilinearly interpolated (usually on GPUs) and have values that are between 1 and 2. Should the assigned color be between red and green - or either red or green? This is controlled by `interpolation_mode`, which either can be `nearest` or `linear`; `wsi_heatmaps` will use `linear` as default.

Here is an example of a simple color LUT specification that is used for heatmap display (note: color entries are truncated) and should be stored in `wsi_presentation/colorluts`:

```
[  
    {  
        "name": "RocheTumor"  
        "textgui": "RocheGUI",  
        "colorspace": "rgba",
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
"interpolation": "linear",  
  
"multiplier": 1.0,  
  
"offset": 0.0,  
  
"maxbin": 1.0,  
  
"entries": [  
    0.0,0.26,0.00,0.32,0.00,0.26,0.01,0.33,0.01,0.27,0.01,0.34,"..."  
]  
}]
```

The above example introduces several new concepts:

1. The LUT will be named **"RocheTumor"** and can be referred to by any heatmap or mask that needs a color assignment.
2. If the LUT name is presented in a clinical application, the text specified by **textgui** will be used. In this case, the string **"RocheGUI"** is targeted for remapping via a translation layer.
3. A multiplier and offset can be used to perform contrast/brightness-like operations without having to redefine the actual LUT entries. If not specified, the offset is assumed to be zero and the multiplier to be one (i.e., a NOP).
4. An optional **colorspace** can be specified. As of now, only the RGB and RGBA color spaces are supported, but we'd like to support additional color spaces in the future.
5. There are different ways to specify the bins in the color LUT. In this case, we specify the maximum value associated with the color LUT by the **maxbin** field; the **entries** field then specifies all RGBA values.

Note that this example only specifies the color LUT **"RocheTumor"**, it does not specify which heatmap uses that color LUT. This is specified in **wsi_presentation/presets**, to be presented later in this document.

Marker shape specifications: **wsi_presentation/marker_shapes**

Markers identifying nuclei on the WSI image can be rendered dynamically by combining the cell information (located in **wsi_cells**) and marker

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

specifications (in `wsi_presentation/marker_shapes`). Here is an example of a marker shape specification (inspired by [ArcGIS](#)):

```
{  
  "blk": { "style": "circle", "size": 7, "color": "rgba(0,0,0,255) " },  
  "red": { "style": "circle", "size": 7, "color": "rgba(255,0,0,255) " },  
  "blk_cvd": { "style": "triangleup", "size": 7, "color": "rgba(0,0,255,255) " }  
}
```

The above JSON code fragment specifies two marker types. A few comments:

1. DIPLOMAT supports seven predefined marker styles: circle, triangleup, triangledown, plus, diamond, crossx and square. It is allowed to add other marker styles (like "thinplus", but these will not have to be supported by the overlay generator. If not specified or the specified style is unknown, the marker style generated by the overlay generator should default to square or circle.
2. The size is specified in WSI pixels; it is the circle diameter or the longest shape axis in case of other marker types. It is recommended to specify odd marker sizes as they usually look better in typical implementations. If no size is specified, it is up to the overlay generator to select an appropriate size.

Vertex styles description

`wsi_presentation/vertex_styles` enumerates how vertices should be drawn (vertices are used for cell contours, annotations, and other features). For example, if we want to draw cell contours in pure green with a line thickness of 2 pixels filled in with a very low opacity, this would be an appropriate specification:

```
{  
  "GreenLine2px":  
  {  
    "outline": { "color": "Green,70%", "width": 2 },
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
        "fill": { "Green,10%" }  
  
    }  
  
}
```

Note that the specification for line width and fill color are currently ignored by most viewers. In general, fills for cell contours should be avoided as they obscure the underlying data; usually, "hollow" contours are often preferred by pathologists.

Rendering wsi_cells contents: wsi_presentation/markers

The information in `wsi_cells` and `wsi_presentation/marker_shapes` can now be used to render marker overlays dynamically. This is specified in JSON in the group `wsi_presentation/markers`.

This is how the marker definition would look like for a Roche algorithm that labels nuclei (like Roche's ER algorithm):

```
[  
  
    {  
  
        "textgui": "marker_default",  
  
        "active": true,  
  
        "data": [  
  
            {  
  
                "label": 0,  
  
                "name": "yel",  
  
                "textgui": "tc+",  
  
                "locked": true,  
  
                "visible": true  
  
            },  
  
            {  
  
                "label": 1,
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
        "name": "blk",  
  
        "textgui": "tc-",  
  
        "locked": true,  
  
        "visible": true  
    },  
    {  
  
        "label": 2,  
  
        "name": "cyn",  
  
        "textgui": "im+",  
  
        "locked": true,  
  
        "visible": false  
    },  
    {  
  
        "label": 3,  
  
        "name": "blu",  
  
        "textgui": "other",  
  
        "locked": true,  
  
        "visible": false  
    },  
    {  
  
        "label": 4,  
  
        "name": "mag",  
  
        "textgui": "artifact",  
  
        "locked": true,  
  
        "visible": false  
    }  
]  
]
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
}  
]
```

Note that the `ws_i_presentation` group consists of an array of presets, where each preset is an array of markers. This JSON code specified a preset named `marker_default` (to be translated based on locale) that should be used for rendering of the markers; only one preset can be active at a time. All the points with label 0 should be drawn as a `yel` shape, which definition can be found in `ws_i_presentation/marker_shapes`. The definition also lists the name that should be displayed by the GUI; the hook `textgui` indicates that the text should be translated if needed. It is possible to specify multiple presets in `ws_i_markers` and by adding `textgui` fields, it is potentially possible to automatically generate GUI hooks that allow for selection of presets. To support this, the `visible` attribute was added; it specifies whether markers are visible or not in the default specification. As information about invisible markers can be relayed to the GUI, it is possible to dynamically generate a GUI that would show the name and associated colors for markers that are not visible by default, but can be made visible. If not present, `visible` is assumed to be set to `true`.

The attribute `locked` is another hook that can be helpful for automatic GUI generation; if the presentation of a marker is locked, it cannot be changed by the user. This can be useful for IVD algorithms that have been FDA approved with specific marker settings; by prohibiting users from changing the marker settings, we would eliminate the potential risk of running the GUI out-of-spec. If not present, the default value for `locked` is `false`.

Here is a `ws_i_presentation/markers` description that can be used for rendering of the DUALish example from the `ws_i_cells` section:

```
[  
  {  
    "textgui": "marker_default",  
    "active": true,  
    "data": [  

```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
{  
  
    "label": 1,  
  
    "name": "blk",  
  
    "textgui": "blk",  
  
    "locked": true,  
  
    "visible": true  
  
},  
  
{  
  
    "label": 2,  
  
    "name": "red",  
  
    "textgui": "red",  
  
    "locked": true,  
  
    "visible": false  
  
} ]  
  
},  
  
{  
  
    "textgui": "marker_cvd",  
  
    "active": false,  
  
    "data": [  
  
        {  
  
            "label": 1,  
  
            "name": "blk_cvd",  
  
            "textgui": "blk",  
  
            "locked": true,  
  
            "visible": false  
  
        }  
  
    ]  
  
}]
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
}
```

```
]
```

This specification for DUALish visualization simply states that coordinates with label 1 need to be drawn according to the shape `blk` (black), while coordinates with label 2 need to be rendered as the shape `red` (red squares with 50% opacity). In this case, we added a preset wrapper around the specification; the alternate preset `marker_cvd` only shows the black (label 1) coordinates.

The information in the `wsi_presentation` group will be used to initialize a dynamic tile generator to generate overlays. While outside the scope of the DIPLOMAT specification, it is the intent that clinical applications can provide alternate marker specifications and “override” the specification embedded in the DIPLOMAT specification. This approach allows for a default “look and feel” (embedded in DIPLOMAT) while giving the user the flexibility to choose other representations.

Rendering of masks: `wsi_presentation/masks`

As described earlier, single label segmentation masks are stored in `wsi_masks/predicted_region_mask_l{pyramid_level}`. We now can provide the recipe on how to render the mask (in `wsi_presentation/masks`), an example:

```
[{ "name": "predicted_region_mask", "textgui": "background", "label": 1,
    "locked": true, "visible": false, "color": "rgba(255, 165, 0, 1.0) ",
    "level_opacity": [0.15,0.2,0.25,0.3,0.35,0.4],
  "level": -1} ]
```

The maskname identifies the mask to be rendered; OSG will look for a mask called `predicted_region_mask` and `predication_region_mask_l[0..9]` to find the mask to draw. In this case, the JSON specification provides information on how to render the mask associated with label 1; also note that we provided the label text using `textgui`, which indicates that a dictionary should be applied. By

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

default, the background mask is not visible; if it is visible, its opacity varies with resolution level.

We can add presets (for example two different colors) as follows:

```
[  
  {  
    "textgui": "default_color",  
    "active": true,  
    "data": [  
      {  
        "name": "predicted_region_mask",  
        "textgui": "background",  
        "label": 1,  
        "locked": true,  
        "visible": false,  
        "color": "rgba(255, 165, 0, 1.0) ",  
        "level_opacity": [ 0.15, 0.2, 0.25, 0.3, 0.35, 0.4 ],  
        "level": -1  
      }  
    ]  
  },  
  {  
    "textgui": "alternate_color",  
    "active": false,  
    "data": [  
      {  
        "name": "predicted_region_mask",
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
        "textgui": "background",

        "label": 1,

        "locked": true,

        "visible": false,

        "color": "rgba(0, 165, 255, 1.0) ",

        "level_opacity": [ 0.15, 0.2, 0.25, 0.3, 0.35, 0.4 ],

        "level": -1

    }

]

}
```

A similar approach is used to handle multi-label masks:

```
[

{

    "textgui":"default",

    "active":true,

    "data":[

{

    "name":"predicted_region_mask_1",

    "textgui": "Pen",

    "color": "rgba(128,255,0,128)",

    "locked": true

    },

{

    "name": "predicted_region_mask_2",
```


TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
        "textgui": "Blurry",

        "color": "rgba(128,128,0,128)",

        "locked": false

    }]

}

}
```

In the above example, the preset for the display was named **default**; note that the resolution level is omitted from the mask name. Similarly to the **ws_i_presentation/cells** specification, the recipes present in the DIPLOMAT file are merely used to configure a dynamic tile generator to generate overlay tiles with clearly defined colors. The viewing application can use different recipes to display the masks in a completely different fashion.

Dynamic rendering of heatmaps: **ws_i_presentation/heatmaps**

To render the heatmaps, we need to combine the heatmaps (**ws_i_heatmaps**) and the associated color LUTs (**ws_i_presentation/colorluts**). Let's first look at the color lookup tables; for Roche's Her2 Dual ISH algorithm, we could define two color lookup tables, the traditional heat LUT and a color assignment called **cividis** which is optimized for people with color deficiencies. The contents of **ws_i_presentation/colorluts**:

```
[

    {

        "name": "heat_lut",

        "textgui": "heat_lut",

        "colorspace": "rgba",

        "interpolation": "linear",
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
"locked": false,

"multiplier": "1.0",

"offset": "0.0",

"entries": {

    "0": "rgba(248,248,248,0) ",

    "1": "rgba(255,255,153,255) ",

    "2": "rgba(255,255,0,255) ",

    "3": "rgba(255,204,0,255) ",

    "4": "rgba(255,153,51,255) ",

    "5": "rgba(255,102,0,255) ",

    "6": "rgba(255,0,0,255) ",

    "7": "rgba(230,0,0,255) ",

    "8": "rgba(205,0,0,255) ",

    "9": "rgba(180,0,0,255) ",

    "10": "rgba(155,0,0,255) ",

    "11": "rgba(130,0,0,255) ",

    "12": "rgba(105,0,0,255) ",

    "13": "rgba(80,0,0,255) ",

    "14": "rgba(55,0,0,255) "

},

"level_opacity": [ 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6 ]

},

{

    "name": "cividis_lut",

    "textgui": "cividis_lut",

    "colorspace": "rgba",

    "interpolation": "linear",
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
"locked": false,

"multiplier": "1.0",

"offset": "0.0",

"entries": {

    "0": "rgba(248,248,248,0) ",

    "1": "rgba(248,231,37,255) ",

    "2": "rgba(205,225,29,255) ",

    "3": "rgba(152,216,62,255) ",

    "4": "rgba(103,204,92,255) ",

    "5": "rgba(64,189,114,255) ",

    "6": "rgba(37,172,130,255) ",

    "7": "rgba(31,153,138,255) ",

    "8": "rgba(36,135,142,255) ",

    "9": "rgba(43,116,142,255) ",

    "10": "rgba(52,97,141,255) ",

    "11": "rgba(61,77,138,255) ",

    "12": "rgba(69,53,129,255) ",

    "13": "rgba(72,28,110,255) ",

    "14": "rgba(68,1,84,255) "

},

"level_opacity": [ 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6 ]

}
```

]

In the example above, we define two lookup tables that will be made available to the application; they are not locked, which means that the viewer application might provide hooks to provide interactive manipulation of the color assignments.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
[  
  
  {  
  
    "textgui": "dualish_heat",  
  
    "active": true,  
  
    "data": [  
  
      {  
  
        "overlay": "DUALish score",  
  
        "name": "Example Name",  
  
        "colorlut": "heat_lut",  
  
        "visible": true,  
  
        "locked": false,  
  
        "level_opacity": [ 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6 ]  
  
      }  
  
    ]  
  
  },  
  
  {  
  
    "textgui": "dualish_viridis",  
  
    "active": false,  
  
    "data": [  
  
      {  
  
        "overlay": "DUALish score",  
  
        "colorlut": "cividis_lut",  
  
        "name": "Example Name",  
  
        "visible": true,  
  
        "locked": false,  
  
        "level_opacity": [ 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6 ]  
  
      }  
  
    ]  
  
  }  
]
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
        }  
    ]  
}  
]
```

Displaying panels: wsi_presentation/panels

Some algorithms only provide marker positions, other algorithms might want to display masks or heatmaps; to avoid hardcoding of algorithm GUIs in Roche products, we tried to provide some hooks in Diplomat that allow algorithms to specify which panels should be created to allow users change the viewer settings. Here is an example of a panel specification:

```
[ { "textgui": "markers_panel", "visible": true, "contents": [ "marker_default" ] },  
  {"textgui":"mask_panel","visible":true,"contents": "masks"}]
```

The example requests two panels, both visible by default, for control of markers and masks.

The marker panel will display contents associated with "marker_default", which happens to be defined in wsi_presentation/markers:

```
[{ "textgui":"marker_default", "active":true, "data":[{"label":1, "shape": "red", "textgui":"tc+",  
          "locked": true, "visible":true },{ "label":4, "shape": "blu", "textgui":"tc-",  
          "locked": true, "visible":true },{ "label":2, "shape": "cyn", "textgui":"ar",  
          "locked": true, "visible":false },{ "label":3, "shape": "blk", "textgui":"im",  
          "locked": true, "visible":false },{ "label":5, "shape": "yel", "textgui":"oth-",  
          "locked": true, "visible":false }]} ]
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

Note that, in turn, shape definitions in the markers can be found in `wsi_presentation/shapes`.

The mask panel simply refers to "masks" (`wsi_presentation/masks`) that would need to be presented:

```
[{ "maskname": "background", "textgui": "background", "label": 1,  
    "locked": false, "visible": true, "color": "rgba(255, 165, 0, 0.4) ", "level": -1 } ]
```

After parsing the `wsi_presentation` entries, OGS creates a JSON encoding that can configure the GUI; that JSON string is usually referred to as "presentation state" and is used to communicate between OGS and clinical viewers. During development, we can manipulate the presentation state JSON to explore alternative shapes, colors, and transparencies of overlays; the next section describes two tools that are used with Roche that are used for this purpose.

Targeted for future release: annotations and scenes

wsi_annotations: Annotations

The typical workflow of whole-slide analysis is to have algorithms analyze the entire WSI; a typical DIPLOMAT file will not contain any (manual) annotations. However, DIPLOMAT does include extensive support for annotations for three main reasons:

1. We need to have a standard annotation format as part of Roche Open Environment - we might as well specify it as part of DIPLOMAT;
2. A very important use-case for algorithm development is to run prototype algorithms using manual annotations as additional input and compare the

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

resulting algorithm scores against the ground truth. In that case, storing the input user-annotations in DIPLOMAT is mandatory as the annotations are an integral part of the algorithm results (cannot be reproduced without these).

3. WSA analysis might automatically generate annotations that identify areas of interest; while these often can be stored as masks, treating these as annotations can be useful.

DIPLOMAT uses GeoJSON as the underlying format to specify annotations; if present, the GeoJSON description should be written to the HDF5 group **ws_i_annotations**. To distinguish between algorithm and user annotations, we use different groups to store these; algorithm annotations are stored in **ws_i_annotations/algorithm**, while user annotations should be stored in **ws_i_annotations/user**.

In the current workflow of DP systems, manual annotations tend to be simple and used to identify diagnostically relevant areas of the WSI. In practice, most manual annotations are limited to simple include and/or exclude regions with labels like "tumor", "exclude", "artifact", etc. However, DIPLOMAT should be able to handle more complex use-cases, notably an approach that can impose a hierarchical organization of the annotations. There are already various approaches (see <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6669998/>), but none of them seems to have great coverage for the pathology laboratory. The approach that DIPLOMAT takes is to store all annotations as individual features in a "flat" organization; every annotation should have a unique ID. Optionally, annotations can contain the ID of its "parent" node which can define an annotation hierarchy if needed; this is usually not done in typical workflows, but DIPLOMAT is flexible enough to support extensive hierarchies.

DIPLOMAT supports several annotation types. These are:

- **Flexpolys**, which are polygons and therefore are stored as a GeoJSON **Polygon**. Every annotation should have an association with a unique label (e.g., "Tumor" or "Stroma") and a unique id (e.g. "3242") - these are mandatory fields. When saved to the backend, flexpolys usually get a unique identifier, creation date, and a bounding box. Optionally, we can have an associated comment (a string), the parent of the flexpoly

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

(in case of hierarchical annotations) and the UUID of the analysis job that generated it (if applicable).

- **FOVs**, which are simple rectangular annotations which are not editable. They have the same fields as Flexpolys, but their GeoJSON geometry type is **MultiPoint** and contain two points (the top-left and bottom-right coordinates of the bounding box).
- **Counter points**, which are point annotations. They have the same fields as Flexpolys, but their GeoJSON geometry is **Point**.
- **Arrows**, which again have the same fields as Flexpolys, but their GeoJSON geometry is **LineString** containing two Points; the arrow will point to the second coordinate.

Note that annotations define geometry of interest, but do not specify how these are rendered by the GUI. For example, a counter point typically is rendered using a circle of a certain radius, fill color, and transparency - but in DIPLOMAT, these properties are stored in a separate HDF5 group called **ws_i_presentation**.

Examples of Flexpoly annotations

As mentioned, annotations are implemented as GeoJSON features. Here is a hypothetical example of an algorithm-generated FOV with an embedded exclusion region:

```
{  
  "type": "FeatureCollection",  
  "features": [  
    {  
      "type": "Feature",  
      "bbox": [80,1000,120,1040],  
      "properties":  
        {  
          "cull_type": "exclude",  

```


TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
"id": "321",

    "parent": "322",

    "label": "pen",

    "uuid": "f4611e6c-d48a-11eb-b8bc-0242ac130003"

},

"geometry": {

    "type": "Polygon",

    "coordinates": [[[80,1000],[80,1040],[120,1040],[120,1000],[80,1000]]]

}

},

{

    "type": "Feature",

    "bbox": [10,500,220,1200],

    "properties":

    {

        "cull_type": "include"

        "id": "322",

        "label": "tumor",

        "uuid": "3beb929e-d48b-11eb-b8bc-0242ac130003"

    },

    "geometry": {

        "type": "Polygon",

        "coordinates": [[[10,500],[10,1200],[220,1200],[220,500],[10,500]]]

    }

}

]
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

The **parent**, and **uuid** fields are optional; the **label** and **id** fields are mandatory to uniquely identify the type of annotation. In this example, the algorithm also added the cull behavior of the polygon (**cull_type**) to encode the default cull behavior of the annotation; it has the possible values **include** or **exclude**. All annotation types basically use the same form, but can be distinguished by the type of geometry used.

Again, hierarchical relationships are optional - as of now, it is not used by Roche DP products. However, we wanted to make sure that DIPLOMAT would be able to store information associated with annotation hierarchies.

User annotations are usually not present in DIPLOMAT files; they only will be included when the WSA was done with user annotations as additional information (which is frequently done using clinical validation of algorithms). In that case, annotations can have additional fields like the **username** (the e-mail address of the user or another unique identifier) and a **timestamp**. If present, user annotations are stored in **ws_i_annotations/user**.

Be careful of the direction of the polygon path! The GeoJSON specification is specific with respect to the direction of the polygon path: any outer path needs to be specified in a counter-clockwise order, while an inner path (i.e. a hole in the polygon) needs to be specified in a clockwise order.

If the cull type field is unknown, DIPLOMAT can determine the culling type based on the direction of the polygon path: clockwise is **"exclude"**, while counter-clockwise is **"include"**. The previous example can also be expressed as follows:

```
{  
  
  "type": "FeatureCollection",  
  
    "features": [  
  
      "type": "Feature",
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
    "bbox": [10,500,220,1200],

    "properties":

    {

        "id": "322",

        "timestamp": "2021-04-23T18:25:44.621Z"

        "label": "tumor",

        "username": "sxyz",

        "uuid": "3beb929e-d48b-11eb-b8bc-0242ac130003"

    },

    "geometry": {

        "type": "Polygon",

        "coordinates": [[[10,500],[10,1200],[220,1200],[220,500],[10,500]]

                        [[80,1000],[120,1000],[120,1040],[80,1040],[80,1000]]]

    }

}

]
```

Instead of explicitly specifying **excluded** regions, we can specify "holes" in the inclusion region by adding the coordinates of a clockwise polygon.

Specification of cull types using cullTypes.json

Let's assume that an algorithm generates annotations for "Stromal" areas. What should its cull behavior be: "include" or "exclude"? The proper answer is, of course, "it depends"; when determining "tumor area", stroma is usually excluded, but when answering "what is the stroma area" it should be included.

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

We can control the default cull behavior of annotations by specifying this in a file named **cullTypes.json** that should be present in the same group as the algorithm results. An example:

```
{
  "exclude":
  {
    "label": [ {"pen"}, {"stroma"} ]
  },
  "include":
  {
    "label": [ {"tumor"} ]
  }
}
```

Determining whether an annotation is to be included or excluded is a three step process:

1. If the cull type is explicitly specified in the annotation, it will always be the specified type.
2. If the cull type is not specified, **cullTypes.json** is used to determine the cull type.
3. If step 2) does not yield an answer, the cull type is based on the polygon orientation (counter-clockwise: include, clockwise: exclude)

Dynamic rendering of annotations

Annotations are usually polygons; they need to be rendered with a specified color and line thickness. As mentioned previously, DIPLOMAT also supports fill color specifications, but we expect most implementations to ignore polygon fills. Here is the generic specification in **wsi_presentation/annotations** for a preset that draws all annotations in green:

```
{
```

TEMPLATE: FT0500-504C

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
    allGreen: {  
        "*": "GreenLine2px"  
    }  
}
```

It should be obvious that the JSON code says "draw any annotations in green lines with a 2 pixel thickness". The specification of **GreenLine2px** can be found in **ws_i_presentation/vertex_styles**.

Using a combination of regular expressions (like "*" - any annotation, "cullType:exclude" - any exclude annotation, "id:332" - the annotation with ID 332), we can fine-tune the annotation display as needed. For example, exclude annotations in red, include annotations in green, and annotation with id 42 in yellow (with a more opaque fill color) can be accomplished as follows:

```
{  
    "GreenRedYellow":  
    {  
        "*": "GreenLine2px",  
        "cullType:exclude": {  
            "Outline": {  
                "Color": "Red",  
                "Width": 2  
            }  
        },  
        "id:42": {  
            "Outline": {  
                "Color": "Yellow",
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
        "Width": 1
    },
    "Fill": {
        "Color": "Yellow,50%"
    }
}
}
```

The intent of DIPLOMAT is to provide a self-describing, easy to understand way to store algorithm results and provide ways to specify a default setting on how they should be displayed. Needless to say, the above example can be easily extended - and abused, like assigning a unique id to automatically generated annotations for the contour of each cell in the WSI, and then specifying a different color to each contour. While DIPLOMAT allows for this, we don't expect applications to render scenes like that with sufficient performance (or at all).

Putting everything together: scenes

In the prequel, we have described how DIPLOMAT stores presets that describe how it describes the way that markers, masks, and heatmaps should be displayed. DIPLOMAT introduces a final concept, scenes, that allows the algorithm results to contain a description on how very complex results should be displayed.

First, let's describe the expected default behavior when a scenes specification is not present. In that case, the rendering order is as follows:

1. heatmap(s)

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

2. mask(s)

3. marker(s)

We first will render the heatmap(s) using the first preset specified in `wsi_presentation/heatmaps`, we then render the mask(s) using the first preset specified in `wsi_presentation/masks`, and finally we render the markers using the first preset specified in `wsi_presentation/markers`.

The individual layers are combined using [alpha compositing](#) using the `over` operator; it is assumed that any RGBA colors are converted to use [premultiplied alpha](#). The blending equation that DIPLOMAT uses for the `over` operator is:

In this equation C_o is the output color, C_a is the input color, and C_b is the color of the new layer.

DIPLOMAT offers support for complex visualizations; its scene description allows complete control on what to display and in which order. With algorithms that use many heatmaps, it is helpful to provide scene definitions that select a subset of the heatmaps and specify which nuclei types and masks should be displayed.

A completely hypothetical example: let's assume we want to display tumor positive and negative cells, overlaid over two masks that contain stroma and background and combine it with three heatmaps called tumorA, tumorB, and tumorC of which tumorA needs to be rendered "under" the other layers.

The order of drawing therefore should be tumorA, background, stroma, cells, tumorC, and tumorB; to make things easier to describe, we assume that appropriate presets have been defined for all masks, cells, and heatmaps.

DIPLOMAT can define a scene in `wsi_presentation/scenes` called `hypothetical` as follows:

```
{  
  
    "hypothetical":
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
{  
  
    TextGUI: "sceneDemo",  
  
    "layers":  
    {  
  
        "heatmaps": "tumorA",  
  
        "masks": "background",  
  
        "masks": "stroma",  
  
        "markers": "default",  
  
        "heatmaps": "tumorC",  
  
        "heatmaps": "tumorB"  
  
    }  
  
}
```

The scene points to a dictionary entry "sceneDemo" which can be used to populate the GUI and provide a description. When selected, the dynamic tile generator should render the layers in the specified order using the presets that are assumed to be stored in the appropriate **ws_i_presentation** groups.

ws_i_scores: Whole slide analysis scores

An algorithm can output an additional slide level score in addition to the pixel based results such as annotations, masks, heatmaps, etc. This score can be used to assess the overall slide with a single metric, which can be

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

categorized, into bins per predefined value intervals. Every score can have additional metadata associated with it.

Values and configuration data for slide level score related keys will reside under the **wsi_scores** group key for each score **score_{#}** e.g. **score_0**, **score_1**, etc.:

- **wsi_scores/score_{#}/value** contains a json dictionary with the score value data consisting of a numeric value

```
{  
  "config": " wsi_scores/score_{#}/score_config",  
  "value": 0.3,  
}
```

- **wsi_scores/score_{#}/stats** (optional) contains a list of json dictionaries with the additional values for slide features related to the wsi score

```
[  
  {  
    "config": " " wsi_scores/score_0/stats_config_0",  
    "value": 0.3,  
  },  
  {  
    "config": " wsi_scores/score_0/stats_config_1",  
    "value": 0.1,  
  }  
]
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

- **wsi_scores/score_{#}/score_config** contains the score configuration data consisting of:
 - **display_name**: a title for the slide level score.
 - **header_name**: a header for the slide level score, i.e. a short description.
 - **score_min**: a minimum numeric value that the score can contain
 - **score_max**: a maximum numeric value that the score can contain
 - **ranges_name**: a string title for the slide level score ranges/bins
 - **unit**: a string unit for the score
 - **ranges**: a list of json dictionaries defining the score bin ranges. Each must contain:
 - **bin_lower_threshold** a numeric lower range threshold for that bin
 - **bin_color** a display color for that bin, in either HEX or RGBA string format
 - **bin_name** a string display name for that bin
 -
- **wsi_scores/score_{#}/stats_config_{#}** contains the score configuration data consisting of:
 - **display_name**: a title for the slide level score.
 - **unit**: a string unit for the score
 - **color**: a display color for that stat, in either HEX or RGBA string format

Example **wsi_scores/score_{#}/score_config**:

```
{
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

```
"wsi_score":{  
  
    "display_name": "Artifact free tissue ratio",  
  
  
  
  
  
  
  
  
    "scoring":{  
  
        "min": 0,  
  
        "max": 100,  
  
        "displayName": "Quality score",  
  
        "ranges":[  
  
            {  
  
                "bin_lower_threshold":80.0,  
  
                "bin_color":" rgba(0, 255, 0, 255)",  
  
                "bin_name":"HIGH"  
  
            },  
  
            {  
  
                "bin_lower_threshold":50.0,  
  
                "bin_color": "rgba(255, 0, 0, 255)",  
  
                "bin_name":"SUFFICIENT"  
  
            },  
  
            {  
  
                "bin_lower_threshold": 0,  
  
                "bin_color": "rgba(0, 255, 0, 0)",  
  
                "bin_level": "POOR"  
  
            }  
  
        ]  
  
    },  
  
    "unit": "%"  
  
}
```

TITLE: D175602 Rev3 uPath 2.2 Interface Control Document (ICD) for Open Environment v1.3

}

Example wsi_scores/score_{#}/stats_config_{#}:

```
{  
    "display_name": "Additional stat",  
    "unit": "mm2",  
    "color": "#FF0000"  
}
```

< End of Document >