# tika-work New Module Tutorial

# File Structure

This document outlines the required files for a new tika-work module as well the internal structure and elements necessary.

All modules must have a separate folder with at one config.js file and the main Javascript source code file.

## A. Configuration File

The file should be called config.js and it exports a javascript object with:

- value:String; module's name
- label:String; module's label
- dataType:String; type of data (image, text, audio, video)
- index:Object {file:String, args:Object}; where file is the relative path from modules folder and args is optional extra param to the module
- priority:Number; order between modules
- modeTypes:Object[] {value:String, label:String, levelTypes: {value:String,label:String}}[]; optional list of modes where value must be "single-type" or "multi-type"
- levelTypes:Object[]{value:String,label:String}[]; optional list of levels of labels where value must be in "single-level", "two-level" or "fixed". Fixed must be include an extra attribute "options:String[]"

Optional:

- showLabelAssigment:Boolean (default: true)
- showLabelExclusivity:Boolean (default: true)

## B. JS Source File

The file should export a React Component Function. The function will receive the parameters described below and the returned component will be display like a tool bar on the left panel.

Parameters that are passed to the module's constructor:

**1. currentTool, setCurrentTool**

Indicates which tool is active in the module. It is an object with name and cursor attributes. Will be used by the corresponding Data Viewer.

`{name: String, cursor: String}`

**2. mouseEvent**

Corresponds to the events fired by the Data Viewer. They vary depending on the type of data and the active viewer. It’s an object with name and type attributes, plus other viewer specifics.

`{name: String, type: String, buttons: Number, point: {x: Number, y: Number}, e: Object}`

**3. toolData, setToolData**

Allows to export data of the current state of the module. Depending on the active viewer, it is aware of this data and update as appropriate. It also allows you to add extra functionalities to the context menu.

`{svg: Object, menuExtra: Object}` for image data type

`{html: Object, menuExtra: Object}` for text data type

`{regions: Object, info: Object, menuExtra: Object}` for audio and video data type

**4. displayMenu(e: MouseEvent)**

Displays the context menu on the active viewer with the labels to select. It is a method that has a MouseEvent as a parameter.

**5. labelSelected**

Changes value when a tag is selected in the context menu. Contains the data related to the selected tag `{fullPath: path String[], label: String, format: Object}`

**6. setFormats(formats: {value: String, label: String, parser: String}[])**

It is a method that must be called when loading the module to indicate the list of formats accepted by the module to export the annotations. Parser must be "json" or "xml".

**7. labelAssignment**

It's a boolean setting field that indicates if the assignment of the labels must be single or multiple.

**8. labelExclusivity**

It's a boolean setting field that indicates if the assignment of the labels must be Exclusive or not.

**9. reset, setReset**

It is a boolean field that changes to true when the annotations of the module must be reset and in that case it must be set back to false.

**10. setUsedLabels(labels: Object[])**

It is a method that indicates the labels that have already been assigned. Must be called when a change occurs with respect to the labels and pass as a parameter the list of used labels.

**11. selectedType**

This field indicates the Type that has been selected in case of working with multi-type. It's an object with the name attribute and others specific for the data type.

`{name: String}`

**12. changeSelectedType(name: String)**

It is a method that allows to change the selected type using the name field

**13. validateAnnotation, setValidateAnnotation**

Used for data import and export. The import gets the data when the metaData attribute changes and returns a response in the runningError and warnings attributes. To export, the module should keep each change updated and set the valid as boolean attribute. If false, it must set the error attribute with the message. For export, for each declared format must assign as an attribute with its corresponding data

**14. labels**

list of available labels. It's a list of Object

**options, setOptions**

object that allows to configure some options of the tools menu

`{_hideFontFormat:Boolean, _hideZoom:Boolean}`

**15. file**

original file uploaded for annotations

**16. filename**

name of file uploaded for annotations. It's an object with name attribute

`{name: String}`

**17. datatype**

string that represents the data type