Release Notes

If this is the first time working with the CCU Toolkit, start out by reading the README.md or README.pdf file. It includes basic information on file layout, and which documentation you should read, depending on what you want to do.

Note: Only changes that require changes to user code or result in changes to behavior are listed here. Other, smaller changes might be listed, but aften are not. So you won't find long lists of bug fixes or documentation improvements.

Updating your CCU Library

Start out by updating your CCU_VERSION variable:

```
export CCU_VERSION=1.0   # Linux or Mac
set CCU_VERSION=1.0      # Windows
```

If you are using Python:

```
pip uninstall ccu
# on Linux/Mac:
pip install https://artifactory.sri.com/artifactory/cirano-pypi-local/ccu-${CCU_VERSION}-py
# on Windows:
pip install https://artifactory.sri.com/artifactory/cirano-pypi-local/ccu-%CCU_VERSION%-py
```

You will need to login using your Artifactory user name and API Key.

And check that the right library is loaded:

```
pip list | grep ccu   # Linux or Mac
```

If you are using Java:

Download the jar file from Artifactory, either using the artifactory.sri.com web page, or using this curl command (and getting the ARI_KEY from your account apge on the artifactory.sri.com web site):

```
curl -H 'X-JFrog-Art-Api:API_KEY' -O "https://artifactory.sri.com/artifactory/cirano-local/
```

Note: On windows, do not use the single or double quotes in the command line.

Make sure you are using the jar file with the right version in it: `ccu-%CCU_VERSION%.jar` (Windows) or `ccu-$CCU_VERSION.jar` (Linux)

Review your Dockerfile and *.yaml file to the examples here and in the README-Integration document to see if they need changes. **For release 1.0, changes are needed!**

Double check the ccu-config.yaml file in the sandbox to make sure it is still correct for your machine.

Release Notes For 1.0

The goal of this release is to preview the CCU integration library and support tools needed for official evaluation 1. Therefore, it is important to report any bugs that you find as soon as possible so we can get them fixed long before the evaluation.

Items in bold below are required changes and your container will not run with the new library unless they are made. There are only two of these none backward compatible changes:

1. In Dockerfiles, set CCU_CONTAINER to the name of your container and change pip install.
For example:
`ENV CCU_CONTAINER="sri-wav2vec"`
Do not set it to 1 as you did in the past.

2. In Dockerfiles, change pip install.
To install the CCU library, you will need a command similar to this Install the library with this command:

```
# This installs the CCU Python library.
RUN --mount=type=secret,id=netrc,dst=/root/.netrc \
    pip install https://artifactory.sri.com/artifactory/cirano-pypi-local/ccu-1.0-py3-none-
```

And then by adding this argument to your docker build commands: `--secret id=netrc,src=../netrc`

These commands assume you have a netrc file in the directory above your Dockerfiles, and it has login information for Artifactory like this:

```
machine artifactory.sri.com
login your-user-name-in-artifactory
password your-api-key-from-artifactory
```

Or you can add these lines to your dockerfile (which do not need the netrc file at all), but that puts your API key into your source code, which you might not want to do. Anyone who has access to your source code will have access to the CCU Artifactory repo under your name:

```
# This installs the CCU Python library on your container.
RUN pip install https://username:apikey@artifactory.sri.com/artifactory/cirano-pypi-local/c
```

If none of this works in your environment, then you can copy the wheel file onto your local host and then copy it into the container to get started. However, please report this (and all other) bugs to me so I can fix them.

3. All messages which are generated based on other messages already in the system, must have a trigger_id field containing a list of the uuids for these other messages (even if that list contains just one uuid).
   See the example and discussion in the ReferenceDocument to see how this is done, but as a quick summary:

   - Assign trigger_id fields to all messages your container creates.
     The trigger_id field is a list of strings which are uuids for the messages that your container used to create the new message. For example, if your container analysed one message with uuid "1234" in order to create a emotion message, then the emotion message should have a line like this to set the trigger_id field to a list containing one string: `emotion_message['trigger_id'] = [old_message['uuid']]`

     Since the trigger_id field is a list, if your container uses two or three different messages to create one of its own, then all of them should be put in the trigger_id list.

     Since all result messages are based on some kind of other message, all result messages should have a trigger_id field set.

   - Assign start_seconds and end_seconds fields to all messages where they make sense.
     Start_seconds and end_seconds fields are floating point number which are the number of seconds since the conversation has started. They are not a timestamp or a timedate, which are absolute times. TA1 containers should not have to create these times. Instead, they should be in the messages your get from the ASR or MT containers.

     These fields should be set any time a result message is based on interactions which have a start time and an end time, but this is not every result message.

   - If there is other data that is important to describing why or how your container generated a message, then please do add more fields to your message to describe how or why your container generated the message. You should document these new fields in the Confluence "Messages and Queues" page, and mark them as optional if they may not be included or if other containers generating the same message may not include them.

There is a new jsonlcheck.py program will print errors if this is not done. You run it on *.jsonl files created by your container to check that the messages contain the right fields. It doesn't catch every case of a missing trigger_id, but it will get most of them. If it complains about your jsonl files, please fix them. In the future, you will be expected to run this command and fix its complaints before pushing updates of your containers. Example commands are:

```
python jsonlcheck.py --help
# This command will just check provenanace related fields (like trigger_id and container_na
python jsonlcheck.py --provenance output.jsonl
# This command will check that all messages have the right fields (including provenance fie
python jsonlcheck.py output.jsonl
```

4. There are many changes to the *.yaml configuration file. These are not required right now, but will be required starting sometime in mid to late January. Please read the ReferenceDocument.pdf to see a complete list, but some of the required change include:

- A new dockerCompose section.
- A new trainingLanguage field.
- Different format of the input and output sections.

There is also a new yamlcheck.py program which you must run on your *.yaml configuration file before submitting it, to make sure it has the required fields and is in the right format. Look at the example.yaml and example_big.yaml files for examples of *.yaml files for your image.

Other improvements in this release, which are available if you want to use them:

- Created a jsonlfilter.py program to filter and format jsonl files in many different ways.
- Created a jsonlcheck.py program to check message formatting and data provenance in jsonl files.
- Improved base_message to accept arbritrary named arguments which automatically become fields in the newly created message, like this:

```
image_start = CCU.base_message('image_start',
                               container_name=CCU.container_name,
                               container_version=container_version,
                               debug=CCU.debug,
                               ccu_version=CCU.__version__,
                               now=CCU.now())
```

- Fixed major bug in library so that every message should have a useful container_id field. Right now some messages have a useless name there.
- The script.py program now accept file names on the command line, so you no longer need to have a -j option.
- If needed inside a container, you can get the "config:" section of your YAML file with the command: `CCU.container_config()`
- If needed inside a container, you can get your entire YAML file with the command: `CCU.container_yaml()`

There are now three new functions which can help check messages both before you send them and after you recieve them:

- `CCU.check_message(message)` Returns True if the message matches the format expected by its type. By default, it checks that the message has all required fields. If `strict=True` is passed in, then it also checks that no extra (non-required and non-optional) fields are included. It is recommended that you check all messages that you recieve and strictly check all messages that you send.
- `CCU.check_message_contains(message, fields)` Returns True if the message contains the fields in the list of fields which is the second argument. This function should be called before using a message recieved from another container if you are using optional or extra fields. If you only use required fields, then calling `check_message` is suffient.
- `CCU.check_provinance(message)` Returns True if the message contains the correct fields for determining proviance. Generally, this requires these fields: container_name, trigger_id, start_seconds, and end_seconds. check_message calls this function, so you do not need to call both to check a message.

- `CCU.check_known(message)` Returns True if the message type is known to the checking system or False if not.
- Added -q/–queues option to logger.py to selectively log message queues.
- Things to remember when submitting containers.
  - Please version your containers! Tag them with both a version number and with "latest".
  - Please do not version them in their names. Do not have images named xyz-analytic03, rather have xyz-analytic and then tag it with 03 (or whatever the version is).
  - Thanks for including sample output for your test runs, but please remember to send jsonl files, not log files. So make these files with a "python logger.py -j output.jsonl" command, not by redirecting the output of the logger.py program.
  - The name and version in your *.yaml file should match the container you send. I know this is hard to remember, especially for the version, but please try.
  - Please be consistent about naming containers. If you have a container named frobitz-finder, please do not have files called frobitzer.tar.gz or frobitz_tagger.yaml, or send a directory called frobizTag, or anything else.
    These small variation is naming become a big problem when I'm trying to wrangle 17 different containers. (And that is the actual situation for CCU: 17 containers and growing.)
  - **Container architecture must be amd64 (which is a version of x86).** You will get this automatically if you build on Linux, Windows or older Macs. However, on brand new Mac M1s, you will get the wrong architecture. Therefore you must pass the `--platform linux/amd64` option to your docker build command. If you want it to run on both architecture pass this argument, but it will make your images much larger: `--platform linux/amd64,linux/arm/v8` on some machines you might need a different arm version.

Release Notes For 0.9

Some message formats have changed, so take a look at the "Messages and Queues" Confluence page and see if you need to update your analytics. These are sometimes non-backwards compatible changes.

There are more fields in the YAML file so please read the README-Integration documentation, look at example.yaml file, and update your YAML files.

The Java CCU library has changed to allow users to send or receive either Map<String, Object> or JSONObject. Before only Maps were supported.

Fixed script.py so jsonl files can have utf-8 strings which work on Windows machines. It always worked on Linux and Mac, and now Windows as well.

Add utilties to create jsonl files and improved README-Integration to dscribe these programs and also point users to the source2stream4ccu utlities written by Michael Youngblood at PARC.

Added 0.1 msec sleep to logger.py to avoid busy polling loop.

Release Notes For 0.8

Added –jsonl option to logger.py program so that the logger can now save jsonl files for future use.

Added –debug option to script.py to print out more information when a new message is inserted into a queue.

Includes bug fixes for CCU-31 and CCU-32 and bug handling white space in /etc/hostname files.