CAPSS API Proposal

This document captures the initial proposal for implementing an automated API process between the pawnbroker/secondhand dealers and the CAPSS application. The scope of this doc is to provide a potential workflow with a high level solution to facilitate discussions and identify problem areas. The proposals are not finalized and subject to change during further discussions and discovery during planning. This document is only relevant during the planning phase and will be superseded once the actual implementation is in progress.

The automated API for Bulk Upload feature will allow clients to have the option of uploading transaction files via a secured API in addition to the Web Browser UI. The API will expose several end points following a restful model. Areas to be considered include authentication and authorization, end point security, and validations. The existing upload process will not change. Only a new feature will be added.

The Bulk Load API will follow a similar process as the Web Browser Upload. The API will allow for uploads and provide feedback on status to include the errors and successful completion. The files can be uploaded through out the day as needed by the client, or as a single transaction once daily so long as file size limits are not exceeded.

This document intent is to address the API process and not on the client subscription process, the actual existing workflow, or other aspects of the system.

The remainder of the document will discuss some of the technical details.

Overview

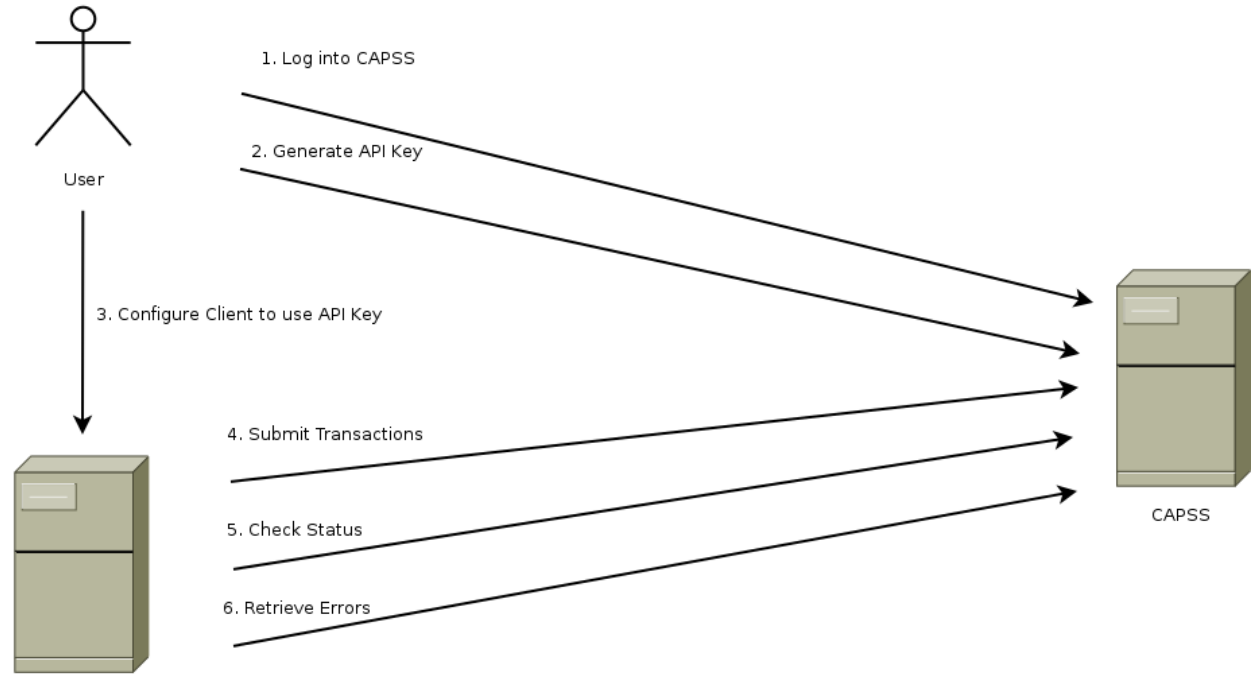
The CAPSS application will provide an integration option supporting a RESTful API exposed to the client following industry standards and OWASP recommendations on security. The intended use case is to allow clients to develop a automated tool to submit transactions in addition to utilizing the web based Bulk Upload. The API is an extension of current web based Bulk Upload Process and simply exposes current behavior for system integration. Several RESTful endpoint will provide the workflow to allow uploading transactions as attachments to basic HTTP methods.

The API will maximize flexibility by providing a limited API that exposes the workflow as a series of endpoints via HTTP methods. Endpoints will allow the user to upload a file, check the processing status, and errors if any. This gives clients the flexibility to implement the services however best suited for them.

Basic Workflow [top]

The CAPSS Bulk Upload API will implement a multistep workflow to the clients that will follow a similar workflow as the online web application Bulk Upload.

- **Registration**: First time users must register for and generate an API Key via the CAPSS web application.
- **Client configuration**: The user will then configure the client application with the key so that it will be presented in all API requests.
- **Bulk Upload Submission**: Submit Transactions with API key
- **Processing Status**: Check For upload completion
- **Submission Feedback**: Retrieve confirmation of successful upload completion
- **Error Retrieval**: Retrieve detected errors.



Architectural Style [top]

The main options for web services are RESTful and SOAP based WebServices. CAPSS API will provides very simple API workflow. A RESTful API is currently favored for it's simplicity and flexibility. SOAP Web Services is a possible option but it will provide a challenge for less technically sophisticated clients and may require more development efforts.

Rest is an architectural style for integrating automated systems. It provides a simple, performant, scalable, and reliable protocol that is platform agnostic. The server presents resource end points to the client, which makes requests to retrieve or modify those resources. The external systems interact with the

API via end point URI's over HTTP using HTTP verbs (GET, POST, PUT, DELETE, etc.). Example include *GET /resource/identifier*. Interactions between the clients and the server are stateless. The proposed resources will be the bulk upload. The actions taken on the resources will be to create the transaction, check the status of the transaction, and retrieve errors for the transaction.

CAPSS API will be written following a RESTful architectural style. The main resource exposed will be the Bulk Upload Transaction. The client will be able to create the transaction, check the status, and retrieve errors. The response will contain status codes that informs the client of the status and the uri's to available for further action.

- **RESTful**
  - Pros
    - Simple to implement
    - Is considered a lightweight implementation
    - data is human readable
    - easy to build
    - Common standard
    - Flexible
    - Maps well with existing HTTP methods
    - A Resource based model
      - Based on entities
      - HTTP verbs
  - Cons
    - No specification
    - Less mature tools
    - No built in validations (type checking)
- **SOAP**
  - Pros
    - Common Standard
    - Runs over HTTP
    - Built in validation (type checking)
    - Mature development tools
    - Existing specifications
    - Discovery protocol for end points
  - Cons
    - Higher Complexity
    - Considered a heavyweight implementation
    - Difficult to develop for less sophisticated users
    - May require more tools, libraries. and software components to implement

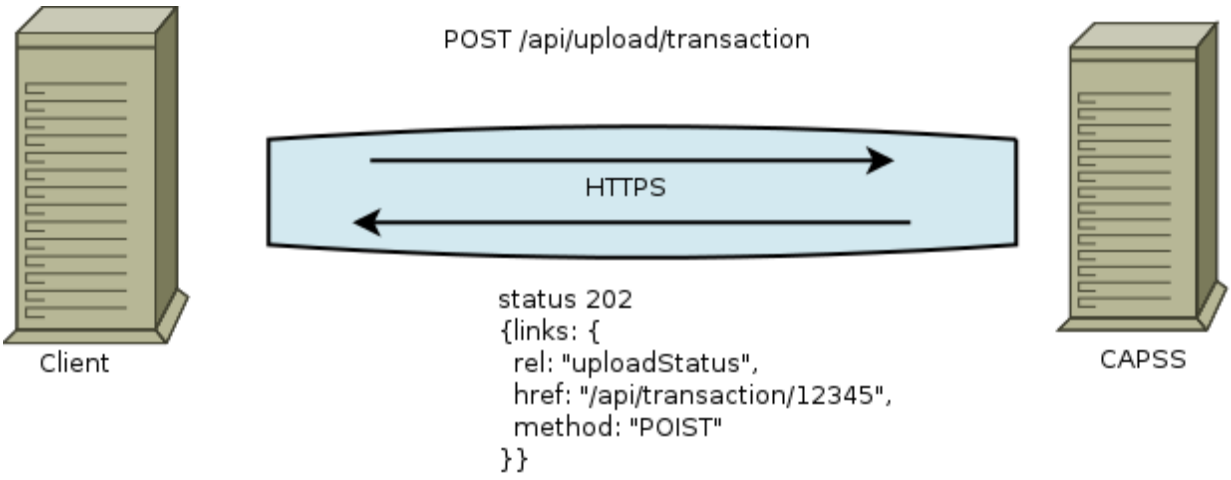
Message Request top

Messages consist of an HTTPS request sent to resource based end points using HTTP Verbs such as GET and POST. The Bulk Upload requests will include an XML transaction file payload similar to the existing version used for online submissions. The CAPSS system will respond with a status code and a JSON response containing available actions with the URI to submit to.

- HTTPS
- HTTP methods (GET, POST, PUT)
- XML File Payload
- Status codes

- JSON response content

The Response will follow an HATEOAS (Hypermedia as the Engine of Application State) architectural style of providing the client with the links available for the next step in the work flow. Example, the response to a bulk upload submission will include the URL that the client will use to check the status of the submitted file. This provides the user with a method of discovery for services.



Responses provide standard status codes to indicate the result of the request. The status are at the HTTP level and not application level. A 200 OK message will be returned if the upload request itself was successful even if the transaction had issues and was rejected for errors. Status codes include:

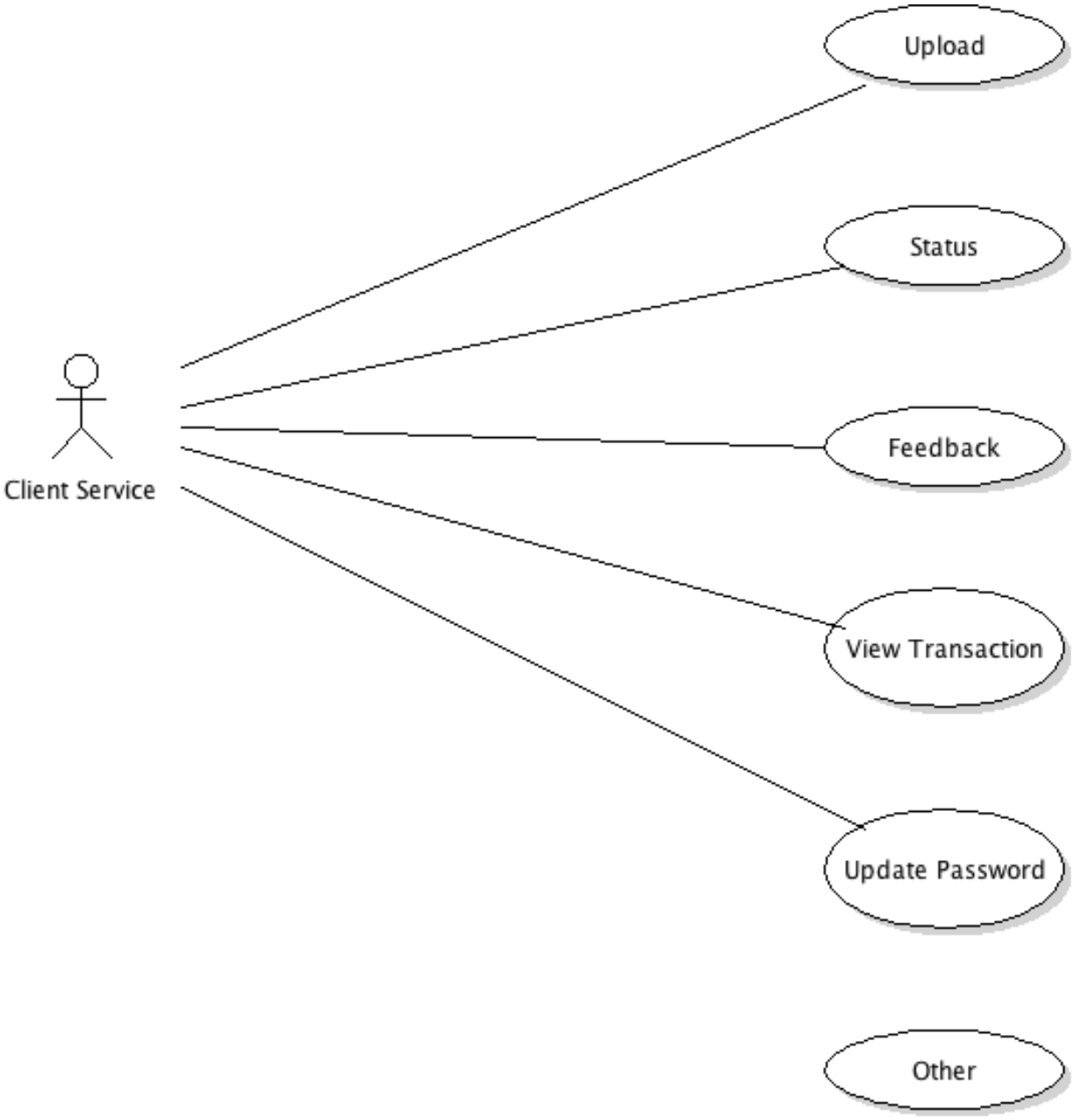
- 200 Ok Successful action. The request was completed successfully
- 400 Bad Request. The request was malformed and hence not processable.
- 401 Unauthorized. Invalid credentials provided
- 403 Forbidden. Account is not authorized for the resource.
- 404 Not Found. An unknown resource was requested.
- 405 Method Not Allowed. Using the wrong HTTP method. Example, DELETE is not supported for any resource.

API Endpoints top

The exact endpoints are to be determined but will follow a RESTful implementation. The Bulk Upload is a resource that will be exposed and various actions can be done on it. The Bulk Upload can be created, status checked, and error feedback retrieved. The following are potential endpoints

- Bulk Upload
  - Upload was valid
  - Upload was accepted for processing
  - Results
    - OK status code returned
    - Error status code returned along with error description
    - URI to check if process is complete

- Processing Status
    - Result
        - Processing is pending
        - Processing is complete
        - URI to check Status
- Submission Feedback
    - Result
        - An OK status code
        - A success message
        - A list of errors
- Transaction Details
    - Optional
    - returns the successful uploaded file



Payloads [top]

The payload will be an XML file complying with the existing CAPSS Schema document modified to allow uploads from multiple stores. Each store must contain a unique identifier registered with CAPSS. Files will be posted using HTTP's mulitpart/form-data

Security [top]

CAPSS will implement industry standard security to provide message integrity and confidentiality as well as securing the app from unauthorized access. HTTPS provides the required level of security to provide trust between the client and server applications, and prevents many common security breaches. Additional security measures will be taken to fill gaps not provided by HTTPS. Authentication will use HTTPS along with a unique identifier.

- Securing communications
- Service Authentication
- Service Authorization
- Token/Password implementation and expiration

Messages [top]

Security will follow Open Web Application Security Project (OWASP) suggestions which outline best practice for implementing web services. Security will be implemented by exposing API endpoints only via HTTPS. HTTPS provides protection for the following security concerns:

- **Message Integrity**: Protects against message tampering .
- **Message Confidentiality** : Protects against eavesdropping
- **Falsified message**: Protects against attacker sending fake messages to client
- **Man-In-the-Middle attack**: Protects against attacker intercepting messages between a conversation.
- **Principal Spoofing**: Protects against attackers using valid credentials pretending to be a legitimate principle.
- **Forged Claims**: Protects against constructing messages with false credentials. Similar to Principle Spoofing
- **Replay of Messages**: Protects a third party from resending previously sent messages.
- **Replay of Message Parts**: Protects against injecting portions of sent messages in new message.

Other security Issues addressed:

- **Denial Of Service Attacks**:
  - protect against repeated request by throttling requests
  - Limiting file sizes
- **Code Execution**:
  - Protect against entity expansion
  - File Validation

Authentication [top]

Three basic options exist for authentication. A token based API key is the preferred method based on it's simplicity and natural fit with web services.. The session is stateless and requires authentication to be done for all requests.

- Token bases API Key
  - The user will register for and embed in all HTTPS requests.
  - Simplest approach
  - Common for APIs
- Username and password
  - User name and password provided as basic auth (secured via HTTPS)
  - Doesn't map well with services
- Certificate based authentication
  - Clients uses a client certificate to submit requests with.
  - Complex and costly to both DOJ and API clients.

Authorization top

Authorization will be performed for each request. Clients will receive the relevant unauthorized code for actions not allowed by service.

Token/password top

Token API key is a common technique to allow client side services to authenticate with the server. The user will generate an API key via a self server registration process from CAPSS. The Key is embedded for every client requests. The client has the option to reset the API key at any time. A high bit secured key complying with industry security standards will be auto generated and supplied to the user.

Passwords/Tokens may expire after 90 days. This feature is still being investigate. If expiration is implemented, the user require a means to reset the password/token. This may be implemented using a manual login process or an automated API.

Testing top

DOJ is considering providing testing functionality for the API to help the clients verify they are integrating. The test API would allow clients to verify

- API is enabled for account
- Security credentials configured correctly
- Reach endpoints
- Submit a document
- Follow workflow.

Corporate/Multi-store accounts top

Stores must be registered as Corporate/multi-store account to upload files from multiple stores. Exact details are to be determined.

Store Ids top

Stores must be identified with a valid ID recognized by the CAPSS application.