**APIs for New Relying Party**

1) When user registers to the RP website the first time, below API will be called:

**URL**: http://10.244.48.148:9090/RelyingParty/public/rpUserRegistration

**Method**: POST

**Input**:

{

“username”: “Jasmira”

“password”: “abc”,

“email”: “[jas@gmail.com](mailto:jas@gmail.com)”,

“phoneNumber”: “9158406756”

}

**Output**:

SUCCESS or FAILURE

**Description**: This API stores the above data entered by user at the time of registrations in the DB and also generate and ID to the user account that is auto incremented. However, no two user with same username are allowed in DB. Username should be unique for every user. Also the Password is stored in encrypted form in DB.

2) After login, when user tries the normal traditional login, below API is fired:

**URL**: http://10.244.48.148:9090/RelyingParty/public/rpUserLogin

**Method**: POST

**Input**:

{

“username”: “Jasmira”

“password”: “abc”

}

**Output**:

SUCCESS or FAILURE

**Description**: This API cross checks the user login. It gets the stored encrypted password from DB based on username passed, decrypts it and checks it against the password passed in the above input JSON and accordingly results into SUCCESS or FAILURE.

3) After user logins into the website using the traditional login, I’m thinking of prefetching the data and keeping it in the browser storage to form input objects for further FIDO related APIs that we will be calling (Ex. FIDO registration, FIDO Authentication and FIDO transaction). This will be done every time the user logs into the website (Either with or without FIDO Login). The reason behind this step is to get the accountid for that particular user, which is auto generated in MYSQL and no information about it is known to the RP website.

**URL**: http://10.244.48.148:9090/RelyingParty/public/getUserDetails

**Method**: POST

**Input**:

{

“username”: “Jasmira”

}

**Output**:

{

“accountid”: “1001”,

“username”: “Jasmira”

“email”: “[jas@gmail.com](mailto:jas@gmail.com)”

}

**Description**: This API gets the user details for given username after the user has successfully logged into the website(either with or without FIDO login), to get the important field “accountid” which is generated in MYSQL and RP website has no information about it.

4) After user logs into RP website, if he wants to do the FIDO registration, below API will be called:

**URL**: http://10.244.48.148:9090/RelyingParty/public/rpFidoRegistration

**Method**: POST

**Input**:

{

"rpDisplayName": "HDFC",

"displayName": "Amogh",

"email": "amogh\_tarcar@persistent.com",

"accountId": "11123"

}

**Output**:

<otp ex.199907>

**Description**: This API calls the FIDO server rpRegRequest API to do the FIDO registration and returns the OTP. We can then use this OTP to create the QR code on the UI.

5) After the Registration is successful, the RP website will poll for FIDO registration status check, for this below API will be called:

**URL**: http://10.244.48.148:9090/RelyingParty/public/fidoRegistrationStatus

**Method**: POST

**Input**:

{

"rpaccountid": "11116"

}

**Output**:

SUCCESS or FAILURE

**Description**: This API calls the FIDO server’s API which polls for registration status check.

6) After FIDO registration, user logs out and tries to login with FIDO. This will cause below API to be fired:

**URL**: http://10.244.48.148:9090/RelyingParty/public/rpFidoAuthentication

**Method**: POST

**Input**:

{

"accountId": "11116",

"rpDisplayName": "PSLBANK"

}

**Output**:

Success

**Description**: This API calls the FIDO server pushNotifyAuthentication API, which push notifies device about user trying to log into RP website using FIDO.

7) After User gives his biometric for authentication, and until the authentication happens at the backend, the RP website will poll for authentication status check. For this, below API will be called.

**URL**: http://10.244.48.148:9090/RelyingParty/public/fidoAuthenticationStatus

**Method**: POST

**Input**:

{

"displayName": "Amogh",

"accountId": "11116"

}

**Output**:

SUCCESS or FAILURE

**Description**: This API calls the FIDO server API that check the Authentication response status.

8) After the Authentication status is returned and the RP website stops polling, the below API needs to be fired to clear the last stored Authentication Response status:

**URL**: http://10.244.48.148:9090/RelyingParty/public/fidoClearAuthentication

**Method**: GET

**Output**:

SUCCESS

**Description**: This API calls the FIDO server API that to clear the last(latest) stored FIDO authentication response status.

**NOTE**: For now, I have designed this API as per my understanding of what is required as input and output for the new RP website to work with FIDO server, with its own backend server in between. Thess APIs might get changed in future as I have not yet started working on the backend stuff for the RP website UI (javascript, ajax calls, etc). Also if more APIs are required, I will add them to this list once they are written in code.