```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import { cloneDeep } from "lodash";
import InternalService from "services/common/InternalService";

export const updateSignatureProcess = createAsyncThunk(
    "Internal/updateSignatureProcess",
    async (data, { rejectWithValue }) => {
        try {
            const { onSuccess } = data;
            const payload = cloneDeep(data);
            delete payload.onSuccess;
            const response = await InternalService.updateSignatureProcess(data);
            if (onSuccess) onSuccess(response.data);
            return response.data;
        } catch (err) {
            return rejectWithValue(err.message || "Error");
        }
    }
);

const initialState = {
    loading: false,
};

export const internalSlice = createSlice({
    name: "Internal",
    initialState,
    reducers: {
        showLoading: (state) => {
            state.loading = true;
        },
        setHisInfo: (state, action) => {
            state.hisInfoList = action.payload;
        },
    },
    extraReducers: (builder) => {
        builder
            .addCase(updateSignatureProcess.pending, (state) => {
                state.loading = true;
            })
            .addCase(updateSignatureProcess.fulfilled, (state, action) => {
                state.loading = false;
            })
            .addCase(updateSignatureProcess.rejected, (state, action) => {
                state.loading = false;
            });
    },
});

export const { showLoading, setHisInfo } = internalSlice.actions;

export default internalSlice.reducer;
```