UX CHECKLIST FOR WALLET DEVELOPERS

We have compiled a checklist of good practices that can be applied to any cryptocurrency wallet with a graphic user interface.

ZCASH FEATURES

If you're building a wallet that supports Zcash, we encourage that you follow these guidelines.

ADDRESSES

☐ Use an address persistently for each use.

Use an address like how you would use different bank accounts (spending, saving, business spending). Transparent addresses aren't private, so we urge users to keep this in mind. We discourage using a new transparent address for each transaction; this only provides a false sense of privacy. Shielded addresses maintain the privacy of transactions, there is no added benefit of using a new shielded address per transaction.

☐ Indicate that transparent addresses are *not* encrypted!

A transaction involving a transparent address (either as sender or recipient) posts the details of the transparent address and amount publicly on the blockchain.

☐ Indicate that shielded addresses *are* encrypted!

A shielded transaction, where funds are sent from one shielded address to another shielded address, only reveals a transaction legitimately and safely happened. The sender, receiver, and amount are not revealed on the blockchain.

☐ If there is no zaddr support, state so clearly.

Most wallets throw an error stating, "invalid address" or "invalid input." We suggest instead saying, "shielded address are not supported." to indicate that shielded addresses are still valid addresses.

☐ Provide taddr and zaddr support if possible.

Shielded addresses provide privacy via encryption and is Zcash's main feature. However, most mobile and desktop wallets don't support sending to shielded addresses, so some users will likely be unable to send ZEC to a shielded address.

☐ Warn users when sending from zaddrs to taddrs (deshielding transactions).

Explicitly tell users that they are about to reveal transaction information. We don't think warnings other types of transactions are necessary.

☐ Show available balance vs owned balance.

Show two balances, one which includes unconfirmed finds, and another not including unconfirmed funds, i.e. "Balance: 621.14321 ZEC (605.35620 ZEC spendable)."

TRANSACTIONS

☐ Clearly state the fee structure:

The default transaction fee is set at 0.0001 ZEC. We encourage that you use the default fee, so that transparent and shielded transactions have the same fee--that way, privacy doesn't cost more for users!

☐ Disable users from setting their own transaction fees.

Do not allow users to customize fees. Our network is fast enough that mining incentivization is not an issue. Unique transaction fees can cause linkability within transactions, especially for zaddrs.

☐ Do not differentiate different types of transactions:

We do not currently distinguish between different types of Zcash transactions: transparent (transparent to transparent), shielding (transparent to shielded), deshielding (shielded to transparent), and shielded (shielded to shielded) transactions. This is complicated by the ability to send to/from a combination of shielded and transparent addresses.

☐ Use the default transaction expiry time.

Transaction expiry (see ZIP here) is set to 20 blocks by default, which is ~1 hour. Use this default global runtime option so Zcash users can develop a consistent expectation of when Zcash transactions expire. We don't support expiry time as a per-transaction runtime option.

☐ Visibly mark newly sent transactions in a "pending" state.

We suggest having a "pending transactions" or "unconfirmed transactions" section, but you can also distinguish it in the list of chronological transactions by using a color or an icon.

☐ Tell the user the expected remaining time to expiry.

Users should be able to see how much time/blocks are remaining until their transaction expires. Once confirmed (10+ confirmations), unmark the sent transaction visibly in a "complete" state.

☐ If expired, visibly mark the transaction expired and and notify the user.

Rather than deleting the attempted transaction, keep the expired transaction in the log, but distinguished as such. We also encourage giving users suggestions on troubleshooting their transaction.

VIEWING KEYS

☐ Use viewing keys for watch only wallets:

Share a viewing key with yourself to create a wallet that tracks your funds while keeping your main funds offline. Watch-only wallets are the first application of viewing keys; we exploring additional use cases as well.

☐ Secure communication channel:

Encourage secure communication channels by supporting one; viewing keys should not be copy and pasted into a text or email.

☐ Indicate that viewing keys are for all incoming transactions:

At version 1.0.14, a viewing key allows the holder of the key to see all incoming transactions since the zaddr was created, but not outgoing transactions.

MEMO FIELDS

☐ Show the memo field in the UI:

Even if the memo field is empty, show that the field is empty rather than removing it from the UI. This is a good nudge to remind users that a memo field exists.

☐ Liberal use:

The memo field is always present and is always exactly 512 bytes long; this is necessary for privacy so that an observer can't detect the different usage patterns and memos. This means that the cost is baked in so that you don't pay a higher fee for including a memo, so encourage users to use the memo.

GENERAL

We encourage that you use this checklist to catch common usability problems before launch or user testing.

USER INTERACTION

☐ Progressive disclosure:

Things should progress from simple to complex; only the necessary or requested information is displayed at a given time. This helps manage complexity without becoming confused, frustrated, or disoriented.

☐ Feedback:

Every action should produce a visible, understandable, and immediate reaction. Failing to acknowledge an interaction can lead to unnecessary repetition of actions or errors (i.e. clicking "send" multiple times).

☐ Priming:

Tell people what they can expect and what they should do. For example, explaining that a camera is needed to scan QR codes before you ask for camera permissions is likely to have users who want that feature to accept it.

☐ Communication:

Be context-aware of what the user is doing and the nature of the message. For instance, notify of events like transaction confirmations with push notifications, since they're probably not waiting on the app for the confirmation.

☐ Error handling:

The best way to handle errors is to prevent them. But if one occurs, put next to the relevant input field (not just at the top or bottom of the screen) to show users what they need to fix without searching for it. It should describe what happened, why it happened, suggest a fix, and not blame the user.

USER INTERFACE

☐ Hierarchy:

Information is presented in order of importance and the visual hierarchy of actions on a screen matches what the user expects to do first, second, third, etc.

☐ Simplification:

Limit the choices that a user is presented with per screen. Provide appropriate filters if there is a large data set.

☐ Consistency:

Components with a similar behavior should have a similar appearance. For example, all buttons that send a transaction should be blue, square, and labeled 'send.'

☐ Predictability:

Set good expectations. From looking at your interface, with no previous use, users should be able to answer things such as "where am I?," "what can I do here?," "where can I go from here?," and "what does this button do?."

☐ Visibility:

Discoverability shouldn't involve luck or chance. If a page requires scrolling, hint that more content is below the screen by showing half of an image. If there are some screens you want users to find, the menu that links to those pages persists everywhere.

NAVIGATION

☐ Persistence.

The navigation bar should always be visible on every screen. If it isn't, users don't know what to do next or don't know how to do the next thing.

☐ Uniformity:

Similarly styled navigational elements should behave similarly. Additionally, elements of navigation should never appear and disappear, rearrange in order, or move to a different location.

☐ Method:

Choose the method that most easily lets the users find what they want. This is specific to the use case. Method include browsing via a navigation system, searching with keywords, or filtering to narrow large lists.

☐ Sorting:

Alphabetical sorting is avoided unless necessitated by many navigational choices (7+). Sort by relevance, related groups, or anything else instead.

☐ Labeling:

Use meaningful labels and icons for navigation menu items, links, and buttons. Don't force people to chase information they need.

CONTENT

☐ Market information:

Provide an up-to-date crypto to FIAT currency conversion, along with current exchange rates between cryptocurrencies.

☐ Network information:

Tell users if their transaction is likely to be processed quickly or not, based on mempool congestion.

☐ Fee information:

Show how much the fee is, what % of the transaction it is, and if it's added on top or included.

☐ Simplify jargon:

Translate what a concept or event affects the user, rather than exposing or explaining what it is technically. For instance, say if the transaction has been confirmed or not, instead showing the number of confirmations or how many confirmations is considered safe.

VISUAL DESIGN

☐ Alignment:

Every element in the UI should be aligned with one or more other elements. Alignment provides cognitive stability and creates visual relationships. In this same vein, left-align large blocks of text as users need to expend more energy to track the lines. Eyes fatigue faster, comprehension slows, but the users may not be aware why.

☐ Proximity:

Group certain elements (navigation, header, articles, etc.) contextually to form a perceived whole. For the same reason, visually separate unrelated items.

☐ Repetition:

Use repetition to create a hierarchy of visual styles. This principle applies to fonts but also colors, textures, and graphical elements. (For instance, all titles should be of one size, all buttons are square, all colors are in a color palette, etc.) Reusing elements of visual styles in visual elements creates cohesiveness.

☐ Contrast:

Text is easily readable when stark, complementary colors are used A lack of contrast between text and background strains the eyes because they don't know which color to focus on.

Let's Talk!
https://chat.zcashcommunity.com

Get updates!
@zcashco

Updated: May 29, 2018