Hello again to the members of the WoWUIDev Discord server from the WoW Team at Blizzard! Midnight Public Alpha begins tomorrow, and so we wanted to give you all an update on what to expect in Alpha in terms of addon APIs.

First, let's get the big questions out of the way:

- The addon API changes we have been talking about for the past several months are coming in Midnight and they will be active in Public Alpha from day 1.
- We will be granting Alpha access to addon developers to help provide us with feedback on these API changes. The list of addon devs maintained by the WoWUIDev mods will be our primary source for these invites. Please check with the server mods if you aren't sure if you're on the list or for consideration to be added.
- Unlike in past expansions, we will have addons enabled throughout Midnight's Alpha and Beta phases.
- We do expect heavy iteration on these systems throughout Alpha and Beta and will do our best to pass along updates on changes as they go out.

OK, now let's dig into the actual addon API changes coming in Midnight (after this very important Disclaimer).

Disclaimer

Everything you read below is subject to change, and while we are confident that the core systems described are close to what will ship with Midnight, the details can and will change as needed.

Overall Focus & Goals

The overall goal of these API changes is to limit addons' ability to perform complex logic based off combat information, and thus optimally solve problems that would otherwise require player thought and coordination. But a secondary goal (almost as important) is to still allow addons to customize the look and feel of the UI (including combat-related UIs). So, while most combat-related APIs have been affected in some way, we have leaned away from restricting APIs entirely or moving more of our UIs into the secure environment where they wouldn't be accessible to addons at all. Instead, we are utilizing a new piece of technology we call Secret Values.

New WoW Lua Construct: Secret Values

The easiest way to think of Secret Values (Secrets) is that they are like black boxes, which contain a Lua value of any type (number, string, boolean, etc) inside them. Insecure (tainted) Lua code can receive Secrets from our APIs and pass those Secrets into certain APIs, but it cannot actually *see* the value that is inside of that box. Attempting to compare a Secret with another value (Secret or not) will result in a Lua error, as will attempting to perform arithmetic on one. Untainted Lua code has no such restrictions and can operate on Secret Values pretty much exactly like non-Secrets. APIs that return Secrets are called out in the Lua API docs, and you can also test if a value is Secret by calling the `issecretvalue` API.

Passing Secret Values into Script Object APIs

As mentioned above, tainted code is allowed to pass Secret Values into certain APIs. This includes our script object APIs, but passing a Secret into a script object API will result in the object being marked Secret. What does it mean for an object to be marked Secret? In the very simplest sense, it means that some of its APIs will begin returning Secret Values, while others will become inaccessible to tainted code.

A simple example of this is the `SetTexture` API. Let's say one of our APIs returns a Secret Value containing a fileID for a spell icon. Addons are allowed to pass that Secret fileID into a Texture's `SetTexture` API, but doing so will result in `GetTexture` now returning a Secret when called. You can test if an object has been marked Secret by calling the `HasSecretValues` API.

Secret Aspects

You may have noticed that we said an object flagged as Secret will return Secret Values from "some of its APIs" instead of "all of its APIs". One of the reasons for that is a script object construct called Secret Aspects (Aspects). You can think of a Secret Aspect as a collection of related APIs on that object. Passing a Secret into any of the APIs tagged with a Secret Aspect will cause the object to be marked with that Secret Aspect, causing all APIs tagged with that Aspect to return Secret Values.

A simple example are the APIs that set and query the shown state of a script object (`SetShown`, `IsShown`, `IsVisible`). All 3 of these APIs are tagged as part of the *Shown* Secret Aspect. So, if you pass a Secret Value into `SetShown`, the object will be marked with the *Shown* Secret Aspect, causing `IsShown` and `IsVisible` to return Secret Values.

There are a number of different Secret Aspects, and they do not share state with each other. So, an object that been marked with the *Shown* Secret Aspect will not return Secrets from APIs that are tagged with the *Alpha* Secret Aspect. APIs and objects can be tagged with more than 1 Secret Aspect. Note that an object being marked with a Secret Aspect will *not* cause it to be marked as Secret overall (and vice versa). You can test if an object has been marked with a Secret Aspect by calling the `HasSecretAspect` API.

Secret Anchors

When a script object is marked as Secret, it is also marked as having Secret Anchors. When an object has Secret Anchors, its anchor & position APIs are no longer accessible to tainted code. Unlike other APIs affected by the object being Secret, Secret Anchors will propagate to other frames that are anchored to this object. So, if frame B is anchored to frame A, and frame A is marked as having Secret Anchors, frame B will also have Secret Anchors. The Secret Anchor state propagates down the entire anchor chain but does not propagate up the chain (if frame A has Secret Anchors and is anchored to frame Z, frame Z won't inherit those Secret Anchors). You can test if an object has been marked as having Secret Anchors by calling the `IsAnchoringSecret` API.

Removing Secrets from an Object

The Secret state and Secret Aspects marked on an object can be cleared by calling the `SetToDefaults` API. Doing so will remove all Secret state (including Aspects) from the object and return the object to its default state (as if it had just been created).

Conditional Secrets

Some APIs only return Secrets under certain conditions. These APIs are marked with special Secret Predicate tags, indicating which conditions they return Secrets under (e.g. `SecretWhenInCombat`). These predicates can also apply to event payloads.

Lua API Documentation

Secret related information has been added to our Lua API documentation files. Every API that can return a Secret is marked as such in the documentation, as is any API that is not allowed to receive Secrets. If an API belongs to a Secret Aspect that is also shown, as are

Secret Predicates. Overall, we have tried to expose as much information as possible about how our APIs interact with Secrets and will continue to add more as we fine tune things.

Changes to Combat Log, Chat, and Addon Communication

Secret Values are not the only changes coming in Midnight. Combat Log Events are no longer available to addons, and messages in the Combat Log chat tab have been converted to KStrings to prevent addons from parsing the information inside them.

There are also some new rules that apply to communication while the player is in an instance. While in an instance, chat messages will be sent to Lua as Secret Values, and addons are not allowed to send communications to other players (either through addon comms or regular chat).

We Want Your Feedback!

As we mentioned at the beginning of this post, we are granting addon devs access to Alpha and Beta, and addons will be enabled throughout both testing phases. This is to allow you as much time as possible to test out the changes, make necessary adjustments to your addons and pass along issues and feedback to us.

We understand that these changes will have a large impact on how WoW addons are developed, and that losing previously available addon functionality is likely to be frustrating for some of you. We would encourage you to keep an open mind and hop on Alpha to start testing things out. We hope that most addons (including combat addons) will still be able to provide players with useful functionality, even if it is in a different manner than before. We want to hear your feedback! Feedback and bugs should be added to the *#addon-restrictions-feedback* channel, which we will be monitoring regularly.

One final note: For the beginning of Alpha, we have taken a very stringent approach in terms of which APIs are permitted to receive Secret Values from tainted code. This is intentional but also shouldn't be viewed as set in stone. The Secret Value system provides us a lot of flexibility to adjust which situations each individual API can and cannot accept Secrets. We will be evaluating every single API throughout Alpha & Beta and making adjustments as we identify addon needs. Your feedback will be vital in guiding exactly which adjustments need to be made. Thanks, and see you on Alpha!