

Achievements Classic
Public API Documentation

Add-on version 1.00 | Public API 1.4.0
Classic Era and TBC Classic

This documentation was generated directly from the current `api/PublicAPI.lua` source and its related API modules. Methods whose names begin with an underscore are internal integration surfaces and are explicitly not part of the public contract.

PDF navigation: chapters are exposed as bookmarks/outlines in the left navigation pane of the PDF reader. A clickable table of contents follows as well.

Table of contents

1. Overview 4

2. Quick start 4

2.1 Extension TOC . . . 4
2.2 Register a module . . . 4
2.3 Register a first achievement . . . 5

3. Module, storage and registration 5

3.1 RegisterModule . . . 5
3.2 Storage layout and scope . . . 5
3.3 Asynchronous registration . . . 5

4. Criteria and triggers 6

4.1 Custom criteria types . . . 6
4.2 Trigger modes . . . 6
4.3 OR criteria . . . 6

5. Categories and achievements 6

5.1 RegisterCategory . . . 6
5.2 RegisterAchievement . . . 7
Criteria definition . . . 7
Achievement chains and meta achievements . . . 7

6. Title rewards 7

7. Progress bars and UI tabs 8

7.1 Summary progress bar . . . 8
7.2 Empty extension tab . . . 8
7.3 Native achievement tab . . . 8
7.4 Categories in empty tabs . . . 8
7.5 Progress bars inside tabs . . . 8

8. Events and refresh 8

9. Handles, errors, and stability 9

9.1 Read-only handles . . . 9
9.2 Validation errors . . . 9
9.3 Protected callbacks . . . 9
9.4 Compatibility rule . . . 9

10. Complete extension example 9

Constants and limits 10

Public ID ranges . . . 11

Reference: global API object 11

Snapshot structures 11

Reference: module handle **12**

Appendix: criteria types **12**

Final notes **16**

1. Overview

The Achievements Classic public API is an extension interface for other WoW add-ons. It allows extensions to add categories, achievements, criteria, progress bars, and UI tabs without directly mutating Achievements Classic internal data structures.

- Global entry point: `_G.AchievementsClassicAPI`
- Current API version: 1.4.0
- Current add-on version: 1.00
- Supported clients: Classic Era and TBC Classic/Anniversary
- Extension progress is persisted only inside a SavedVariables table supplied and owned by the extension add-on.

Only methods and fields without a leading underscore are part of the public contract. Methods such as `_GetTabs()` or `_TriggerExtensions()` are internal integration surfaces and must not be used by extensions.

2. Quick start

2.1 Extension TOC

The extension should load Achievements Classic as a dependency and declare its own SavedVariables table. Persistence scope may be account-wide or per-character.

```
## Dependencies: AchievementsClassic
## SavedVariables: MyAchievementsExtensionDB

Core.lua
```

2.2 Register a module

```
MyAchievementsExtensionDB = MyAchievementsExtensionDB or {}

local API = _G.AchievementsClassicAPI
assert(API, "Achievements Classic is required")
assert(API:IsCompatible(1, 4), "Achievements Classic API 1.4+ required")

local mod = API:RegisterModule("my_extension", {
    name = "My Achievement Pack",
    version = "1.0.0",
    author = "Example Author",
    storage = MyAchievementsExtensionDB,
})
```

`metadata.storage` is required. The table must belong to the extension. AC_LocalData, AC_AccountData, AC_Settings, AC_CharacterPoints, and any table nested inside them are explicitly rejected.

2.3 Register a first achievement

```
local category = mod:RegisterCategory("demo", {
    name = "Demo",
    tab = API.TAB.PLAYER,
    order = 900,
})

local eventType = mod:RegisterCriteriaType("DEMO_EVENT", 1)

local achievement = mod:RegisterAchievement("first_demo", {
    category = category,
    name = "First Demo",
    description = "Trigger the demo event ten times.",
    points = 10,
    icon = "Interface\\Icons\\INV_Misc_Note_01",
    criteria = {
        {
            key = "count",
            type = eventType,
            data = { "sample" },
            quantity = 10,
        },
    },
})

-- Later, when the extension's real game event happens:
mod:Trigger(eventType, { "sample" }, 1, "increment")
```

3. Module, storage and registration

3.1 RegisterModule

`API:RegisterModule(moduleID, metadata)` creates the module handle. `moduleID` must be a non-empty key made of letters, digits, underscore, dot, or hyphen. Registering an existing module ID returns the same handle as long as the same storage table is supplied.

metadata field	Required	Meaning
storage	yes	Extension-owned SavedVariables table
name	no	Display name; default = moduleID
version	no	Version string; default = "unknown"
author	no	Author name

3.2 Storage layout and scope

Achievements Classic creates and manages `storage.AchievementsClassic` below the supplied table, using storage schema 1 and a `modules` subtree. Extensions should not directly mutate this managed subtree.

Whether progress is account-wide or per-character is determined by the extension TOC using `## SavedVariables` or `## SavedVariablesPerCharacter`.

3.3 Asynchronous registration

Registration is finalized in small slices so large extension packs do not block login/reload. A slice performs at most 20 completion checks and uses roughly a 2 ms profiling budget.

- `BeginBulkRegistration()` defers processing and may be nested.
- `EndBulkRegistration()` closes one nesting level and schedules processing when depth reaches zero.
- `IsRegistrationPending()` reports pending bulk work, structure commits, or completion checks.
- `REGISTRATION_COMMITTED` reports the final commit for a module.

```
mod:BeginBulkRegistration()
-- Register many categories/achievements/tabs here.
mod:EndBulkRegistration()

local token = mod:Subscribe(API.EVENT.REGISTRATION_COMMITTED,
    function(module, committedModuleID, changeCount)
        if committedModuleID == module:GetID() then
            print("Registration committed:", changeCount)
```

```
        end
    end)
```

4. Criteria and triggers

4.1 Custom criteria types

`RegisterCriteriaType(key, dataLength, options)` registers an extension-owned type. `dataLength` may be 0 through 16. `options.id` may optionally provide an explicit ID from the public criteria-type range.

```
local typeID = mod:RegisterCriteriaType("MY_COUNTER", 2)

-- A matching criterion might use:
-- type = typeID
-- data = { "zone", 42 }

mod:Trigger(typeID, { "zone", 42 }, 1, "increment")
```

4.2 Trigger modes

mode	Behavior
nil / "increment"	add count to the previous progress
"set"	set progress to count; clamped to 0..quantity

Criteria without `quantity` complete on the first matching trigger. Quantity criteria are clamped between 0 and their required quantity.

`module:Trigger()` uses the same central progression path as the main add-on. Triggering a core criteria type can therefore also advance matching core criteria. Prefer a custom criteria type for extension-only events.

4.3 OR criteria

The internal 0R criteria type is blocked by the public API. Set `anyCriteria = true` on the achievement instead.

5. Categories and achievements

5.1 RegisterCategory

Field	Type	Description
name	string	Required; localized category name
tab	Tab/key/ID	Optional; defaults to player tab
parent	Handle/key/ID	Optional; must belong to the same tab
id	integer	Optional explicit public ID
order	number	Sort order; default 100
unavailable	boolean	Mark category unavailable
featsOfStrength	boolean	API achievement tabs only; main category only; one per tab

Extensions may not create categories or achievements below the native core Feats of Strength category. Extension Feats of Strength categories are allowed only inside an extension-owned API achievement tab.

5.2 RegisterAchievement

Field	Required	Meaning
category	yes	Category handle, module key, or numeric category ID
name	yes	Name
description	yes	Description; empty string is allowed
points	no	Non-negative number; default 0
icon	yes	Texture path, icon name, or file ID
criteria	yes	Non-empty criteria array
id	no	Explicit public ID
hideCriteriaUI	no	Hide all criteria in the UI
anyCriteria	no	Any one criterion completes the achievement
previous	no	Previous achievement from the same extension module
faction	no	"Horde" or "Alliance"
rewardText	no	Reward text
titleReward	no	Title definition
unavailable	no	Unavailable
priority	no	Numeric priority

Criteria definition

Field	Required	Meaning
key	yes	Stable module-local key
type	yes	Core type name/ID or custom type key/ID
data	depends	Address data; value count must match type arity
quantity	no	Target amount > 0
name	no	Criteria label
id	no	Explicit public ID
hidden	no	Hide this criterion in the UI
quantityFormatter	no	function(current, required) -> string

Achievement chains and meta achievements

`previous` may reference only an achievement owned by the same extension module. Completing a later achievement recursively completes still-open previous achievements from that module. Meta achievements may use `COMPLETE_ACHIEVEMENT`; dependent meta achievements are selectively rechecked when the referenced achievement completes.

6. Title rewards

An achievement with `titleReward` also requires a non-empty `rewardText`. `titleReward = true` uses defaults; alternatively supply a table.

Field	Meaning
key	Optional stable title key
format	Male/default format string
femaleFormat	Optional female format string
nativeMask	Optional positive Blizzard title mask
nativeMaskFemale	Optional female title mask

```
rewardText = "Title: the Explorer",
titleReward = {
```

```
    key = "EXPLORER",
    format = "%s the Explorer",
    femaleFormat = "%s the Explorer",
}
```

7. Progress bars and UI tabs

7.1 Summary progress bar

`RegisterSummaryProgressBar()` adds an extension bar to the summary. At most four public extension summary bars may be registered globally.

Field	Meaning
label	Required
category	Optional; recursive progress is counted automatically from available achievements
getProgress	function(module, context) -> current, maximum, text; required when category is absent
isVisible	function(module, context) -> boolean
tooltip	string or function(module, context) -> value
onClick	function(module, button)
order	Sort order; default 100

7.2 Empty extension tab

`RegisterTab()` creates a freely rendered tab. Definition fields: required `name`; optional `title, icon, tooltip, width, order, createFrame, summaryName, onShow, onHide, onRefresh`. Width must be 80-160 pixels, default 115.

Callback signatures: `createFrame(module, parent)`; `onShow/onHide/onRefresh(module, frame)`; tooltip function `tooltip(module)`.

7.3 Native achievement tab

`RegisterAchievementTab()` creates an extension achievement tab rendered by the native achievement UI. `createFrame` is not allowed. Categories are added with `RegisterCategory()` using the tab handle. A maximum of 10 main categories is supported per API achievement tab.

7.4 Categories in empty tabs

`RegisterTabCategory()` is available only for empty tabs. `name` and `createFrame` are required; `parent, order, onShow, onHide, onRefresh, tooltip` are optional.

7.5 Progress bars inside tabs

`RegisterTabProgressBar()` supports up to 10 bars per tab and creates a main category as the navigation target. In achievement tabs progress is calculated automatically from the category, so `getProgress` and `createFrame` are not accepted. Empty tabs require both.

8. Events and refresh

`module:Subscribe(eventName, callback)` registers a listener. The callback automatically receives the module handle as its first argument. `Unsubscribe(token)` removes it.

Event	Callback arguments after module	Meaning
PROGRESS_CHANGED	criteriaType, data	Progress changed
ACHIEVEMENT_COMPLETED	achievementID, sourceModuleID?	Achievement completed; sourceModuleID is present for extension completions
TITLE_CHANGED	selectedTitle	Selected add-on title changed

Event	Callback arguments after module	Meaning
DATABASE_CHANGED	kind, objectID, moduleID, changes, phase	Registration/database change; extension batches use kind="batch"
REGISTRATION_COMMITTED	moduleID, changeCount	A module registration fully committed

```
local token = mod:Subscribe(API.EVENT.ACHIEVEMENT_COMPLETED,
    function(module, achievementID, sourceModuleID)
        print("Completed:", achievementID, sourceModuleID)
    end)

-- Later:
mod:Unsubscribe(token)
```

`RequestUIRefresh()` publishes a targeted extension UI refresh for the calling module. Use it when extension-owned UI state changes without new registration work.

9. Handles, errors, and stability

9.1 Read-only handles

Registration handles are read-only. Assigning fields raises an API error. Category handles include `id, key, moduleID, tabID, kind`; achievement handles additionally include `blizzardID, categoryID, criteriaIDs`. Tab handles include `key, moduleID, kind, tabType, tabID`.

9.2 Validation errors

Invalid registrations raise Lua errors prefixed with `AchievementsClassicAPI:`. Examples include invalid keys, wrong value types, ID collisions, protected Feats of Strength targets, and missing required fields.

9.3 Protected callbacks

Extension UI callbacks are executed through `pcall`. Errors are reported to the WoW error handler together with module ID and callback surface rather than escaping uncaught through the UI path.

9.4 Compatibility rule

Check at least `API:IsCompatible(1, 4)` before using the API. A major-version change is incompatible; higher minor versions within the same major are compatible when they meet your required minimum minor.

10. Complete extension example

The compact example below demonstrates extension-owned SavedVariables, bulk registration, a custom achievement tab, a category, a custom trigger type, and an API event subscription.

```
MyPackDB = MyPackDB or {}
local API = assert(_G.AchievementsClassicAPI)
assert(API:IsCompatible(1, 4))

local mod = API:RegisterModule("example_pack", {
    name = "Example Pack",
    version = "1.0.0",
    storage = MyPackDB,
})

mod:BeginBulkRegistration()

local tab = mod:RegisterAchievementTab("pack_tab", {
    name = "Example",
    title = "Example Achievements",
    width = 115,
    order = 500,
})

local category = mod:RegisterCategory("general", {
    name = "General",
    tab = tab,
    order = 10,
})

local typeID = mod:RegisterCriteriaType("TOKEN_COLLECTED", 1)
```

```
mod:RegisterAchievement("collector", {
    category = category,
    name = "Collector",
    description = "Collect 25 example tokens.",
    points = 10,
    icon = "Interface\\Icons\\INV_Misc_Coin_01",
    criteria = {
        {
            key = "tokens",
            name = "Example tokens",
            type = typeID,
            data = { "default" },
            quantity = 25,
        },
    },
})

mod:RegisterTabProgressBar(tab, "general_progress", {
    name = "General",
    label = "General progress",
    order = 10,
})

mod:EndBulkRegistration()

-- Call from your real game-event handler:
local function OnTokenCollected(amount)
    mod:Trigger(typeID, { "default" }, amount or 1, "increment")
end

mod:Subscribe(API.EVENT.REGISTRATION_COMMITTED,
    function(module, moduleID, changeCount)
        if moduleID == module:GetID() then
            print("Example Pack ready; changes:", changeCount)
        end
    end)
```

Localize extension-owned text inside the extension add-on. The public API stores the supplied strings as-is and does not impose a specific locale architecture.

Constants and limits

The global API object exposes stable constants for tabs, events, client flavors, limits, storage policy, public ID ranges, and every registered criteria type.

Constant	Value / meaning
API.API_VERSION	"1.4.0"
API.TAB.PLAYER	Player/achievement tab
API.TAB.GUILD	Guild tab
API.TAB.STATS	Statistics tab
API.TAB_TYPE.EMPTY	"empty"
API.TAB_TYPE.ACHIEVEMENTS	"achievements"
API.LIMITS.SUMMARY_PROGRESS_BARS	4 extension summary bars globally
API.LIMITS.TAB_PROGRESS_BARS	10 per tab
API.LIMITS.ACHIEVEMENT_TAB_MAIN_CATEGORIES	10 main categories per API achievement tab
API.LIMITS.REGISTRATION_CHECKS_PER_SLICE	20 completion checks per registration slice
API.LIMITS.REGISTRATION_TIME_BUDGET_MS	2 ms time budget per registration slice
API.STORAGE.SCHEMA_VERSION	1
API.STORAGE.ROOT_KEY	"AchievementsClassic"
API.STORAGE.POLICY	"extension-owned-only"

Public ID ranges

Kind	First ID	Last ID
Category / Kategorie	1,000,000	99,999,999
Achievement / Erfolg	100,000,000	999,999,999
Criteria / Kriterium	1,000,000,000	1,999,999,999
Criteria type / Kriterientyp	10,000	99,999

By default IDs are deterministically derived from `moduleID + kind + key`. Keeping the module ID and key stable keeps the generated ID stable. If a collision occurs, supply an explicit ID from the relevant public range or change the key.

Reference: global API object

Method	Returns	Description
GetVersion()	major, minor, patch	Returns 1, 4, 0.
GetAddonVersion()	string	Reads the add-on version from the TOC (currently 1.00).
GetClientFlavor()	string	CLASSIC_ERA or TBC_ANNIVERSARY.
GetClientInterfaceVersion()	number	Current Blizzard interface number for the client.
IsClassicEra()	boolean	Classic Era detection.
IsTBCAnniversary()	boolean	TBC client detection.
IsCompatible(requiredMajor, minimumMinor)	boolean	Major must match exactly; API minor must be at least minimumMinor.
RegisterModule(moduleID, metadata)	module handle	Registers or returns a module. metadata.storage is required.
GetModule(moduleID)	module handle | nil	Returns an already registered module.
GetRegisteredModules()	array	Snapshots with id, name, version, author.
GetCategory(categoryID)	table | nil	Read-only category snapshot.
GetAchievement(achievementID)	table | nil	Read-only achievement snapshot including criteria metadata.
GetEarnedTitles()	array	Earned add-on titles as snapshots.
GetSelectedTitle()	table | nil	Currently selected add-on title.
IsAchievementCompleted(achievementID)	boolean	Local completion state for a core or extension achievement.
GetCriteriaProgress(achievementID, criteriaID)	number	Current local criteria progress.
GetRegisteredSummaryProgressBars()	array	Public summary of registered summary bars.
GetRegisteredTabs()	array	Public summary of registered extension tabs.
IsExtensionAchievement(achievementID)	boolean	True if the achievement was registered through the public API.

Snapshot structures

`GetCategory()` returns: `id, name, parentID, tabID, order, featsOfStrength, available`.

`GetAchievement()` returns: `id, blizzardID, name, description, points, icon, rewardText, titleReward, categoryID, tabID, available, hideCriteriaUI, criteria`. Each criteria snapshot contains `id, name, type, quantity, hidden`.

Title snapshots contain `key, label, format, femaleFormat, nativeMask, nativeMaskFemale, achievementID, sourceModuleID`.

Reference: module handle

Method	Returns
GetID()	string
GetMetadata()	table
BeginBulkRegistration()	self
EndBulkRegistration()	self
IsRegistrationPending()	boolean
RegisterCriteriaType(key, dataLength, options)	criteria type ID
RegisterCategory(key, definition)	category handle
RegisterAchievement(key, definition)	achievement handle
RegisterSummaryProgressBar(key, definition)	read-only handle
RegisterTab(key, definition)	tab handle
RegisterAchievementTab(key, definition)	tab handle
RegisterTabCategory(tab, key, definition)	tab-category handle
RegisterTabProgressBar(tab, key, definition)	category handle
RequestUIRefresh()	nil
Trigger(criteriaType, data, count, mode)	boolean changed
Subscribe(eventName, callback)	subscription token
Unsubscribe(token)	boolean
GetCategoryID(key)	number | nil
GetAchievementID(key)	number | nil
GetCriteriaID(achievementKey, criteriaKey)	number | nil
GetCriteriaTypeID(key)	number | nil
HasSummaryProgressBar(key)	boolean
HasTab(key)	boolean
GetTabHandle(key)	handle | nil
HasTabProgressBar(tab, key)	boolean
IsAchievementCompleted(value)	boolean
GetCriteriaProgress(achievement, criteria)	number
ResetProgress()	nil

`ResetProgress()` deletes all persisted progress for this extension module and reinitializes its registered achievements/criteria. It does not touch other modules or core progress, but it is destructive within the calling module.

Appendix: criteria types

`API.CRITERIA_TYPE` contains the following stable type names and IDs. “Data values” is the number of values used from the `data` array to address a trigger. The exact semantic meaning of those values is type-specific and defined by the relevant core tracker.

For extension-owned events, `module:RegisterCriteriaType()` is the safest choice. Manually triggering a core type can also advance matching core Achievements Classic criteria. `OR` is explicitly unavailable to public achievement criteria; use `anyCriteria = true` for OR semantics.

ID	API.CRITERIA_TYPE.*	Data values
1	KILL_NPC	1
2	KILL_NPCS	1
3	COMPLETE_QUEST	1

ID	API.CRITERIA_TYPE.*	Data values
4	COMPLETE_QUESTS	0
5	OBTAIN_ITEM	1
6	KILL_PLAYERS	0
7	KILL_PLAYER_OF_RACE	1
8	KILL_PLAYER_OF_CLASS	1
9	ATIESH	0
10	REACH_PVP_RANK	1
11	NOT_WORKING	0
12	REACH_PROFESSION_LEVEL	2
13	REACH_MAIN_PROFESSION_LEVEL	1
14	REACH_SECONDARY_PROFESSION_LEVEL	1
15	CRAFT_ITEM	1
16	REACH_REPUTATION	2
17	REACH_ANY_REPUTATION	1
18	EXPLORE_AREA	1
19	REACH_LEVEL	1
20	COMPLETE_ACHIEVEMENT	1
21	BANK_SLOTS	0
22	LOOT_QUEST_GOLD	0
23	BATTLEFIELD_STAT_MAX	2
24	BATTLEFIELD_SCORE_MAX	2
25	BATTLEFIELDS_SCORE	1
26	BATTLEFIELD_WINS	1
27	GEAR_QUALITY	2
28	FISH_AN_ITEM	1
29	LEARN_PROFESSION_RECIPE	2
30	LEARN_PROFESSION_RECIPES	1
31	DUELS	0
32	BATTLEFIELD_MAX_LEVEL_PARTICIPATION	0
33	EMOTE	2
34	BATTLEFIELD_FAST_WIN	1
35	BATTLEFIELD_STAT	2
36	SPECIAL	1
37	BOSS_WITHOUT_MOBS	1
38	BOSS_WITH_ALL_ALIVE	1
39	OR (internal OR)	0
40	KILL_NPC_HEROIC	1
41	ALTERAC_VALLEY_MINE_CAPTURE_MAX	0
42	BG_EYE_GLORY	0
43	BG_EYE_BERSERK	0
44	BG_POINTS	3
45	P3_FIRST_WEEK	1
46	FISH_ANY_ITEM	1
47	HAS_BUFF	1

ID	API.CRITERIA_TYPE.*	Data values
48	KILL_ANY_NPC	0
49	COMPLETE_DAILY_QUEST	1
50	COMPLETE_DAILY_QUESTS	0
51	ARENA_WIN	0
52	ARENA_MAP	1
53	ARENA_RATING	1
54	ARENA_5V5_SURVIVOR	0
55	ARENA_HOT_STREAK	0
56	ARENA_HOTTER_STREAK	0
57	FISH_FROM_POOL	1
58	FISH_UP_BOSS	1
59	COMPANION_PETS	0
60	LOOT_MONEY	0
61	MOUNTS	0
62	READ_BOOK	1
63	DRINKS	0
64	FOODS	0
65	CLASSIC_FOODS	0
66	TBC_FOODS	0
67	TABARDS	0
68	SURVIVE_LONG_FALL	0
69	ALTERAC_BOTH_MINES_WIN	0
70	ALTERAC_KILL_MOUNTED_FIELD_OF_STRIFE	0
71	ALTERAC_VALLEY_ALL_STAR	1
72	ALTERAC_LOYAL_DEFENDER	1
73	ALTERAC_PERFECTION	1
74	ARATHI_MATCH_STAT	1
75	ARATHI_BASE_HONOR_KILLS	1
76	ARATHI_TERRITORIAL_WIN	0
77	ARATHI_RESILIENT_WIN	0
78	SPECIFIC_COMPANION_PET	2
79	OLD_SCHOOL_RIDE	0
80	SOUL_OF_IRON_LEVEL	0
81	SOUL_OF_IRON_BOSS	1
82	SOUL_OF_IRON_NAXX_IMMORTAL	0
83	HOME_ALONE	0
84	WARSONG_MATCH_STAT	1
85	WARSONG_MATCH_CONDITION	1
86	WARSONG_FLAG_CARRIER_KILLS	0
87	TIMED_ITEM_USES	1
88	CAST_SPELL	1
89	LOOT_ITEM	1
90	NOBLEGARDEN_BLUSHING_BRIDE	0
91	GNERD_HONORABLE_KILLS	0

ID	API.CRITERIA_TYPE.*	Data values
92	WINTERVEIL_TIS_SEASON	0
93	WINTERVEIL_MISTLETOE_BROTHER	1
94	PVP_BG_WRECKING_BALL	0
95	PVP_BG_DAMAGE_CONTROL	0
96	PVP_HUG_DEAD_ENEMY	0
97	PVP_WAR_MOUNT	0
98	PVP_CITY_DEFENDER	0
99	PVP_BG_GRIM_REAPER	0
100	DUAL_TALENT_SPECIALIZATION	0
101	EYE_BASE_HONOR_KILLS	1
102	EYE_STORM_GLORY	0
103	EYE_FLAG_CARRIER_KILLS	0
104	EYE_DOMINATION_WIN	0
105	EYE_TARGET_BERSERKER_KILL	0
106	USE_ITEM	1
107	BREWFEST_DRUNKEN_STUPOR	0
108	KNOWN_SPELL	1
109	NATIVE_TITLE	1
110	SERVER_ONLY_FEAT	1
111	CHILDREN_WEEK_PVP_ACTION	1
112	CHILDREN_WEEK_DAILY_QUESTS	0
113	DAILY_QUEST_STREAK	0
114	MIDSUMMER_POLE_DANCE	0
115	WINTERVEIL_RACER_CRASHES	0
116	WINTERVEIL_OGRILA_REINDEER	0
117	LOOT_ROLL_100	2
118	OWN_ITEM	1
119	WEAPON_SKILLS_AT_LEVEL	1
120	TRIAGE_BANDAGE	1
121	TARGETED_ITEM_USE	3
122	KILL_PLAYER_IN_AREA	1
123	TIMED_SOLO_QUEST	2
124	EQUIP_ITEM	1
125	PVP_KILLING_BLOWS	0
126	ALTERAC_KILLING_BLOWS_TOTAL	0
127	PROFESSION_CRAFT	1
128	ALCHEMY_CRAFT	1
129	GATHER_PROFESSION	1
130	PROFESSION_SOURCE_ITEM	2
131	PROFESSION_SOURCE_ITEM_COUNT	2
132	PROFESSION_CRAFT_RECIPE_COUNT	2
133	PROFESSION_UNIQUE_OUTPUT_ITEMS	1
134	PROFESSION_CRAFT_OUTPUT	1
135	PROFESSION_CRAFT_QUALITY	2

ID	API.CRITERIA_TYPE.*	Data values
136	PROFESSION_CRAFT_SLOT	2
137	ENCHANT_UNIQUE_SPELL_USED	0
138	ENCHANT_TARGET_SLOT	1
139	ENCHANT_TARGET_QUALITY	1
140	ENCHANT_TARGET_OWNER	1
141	ENCHANT_TARGET_OWNER_QUALITY	2
142	ENCHANT_TARGET_OWNER_SLOT	2
143	ENCHANT_TARGET_ITEM_LEVEL	1
144	DISENCHANT_TARGET_QUALITY	1
145	DISENCHANT_TARGET_SLOT	1
146	DISENCHANT_TARGET_ITEM_LEVEL	1

Final notes

This document describes the public contract of API version 1.4.0 in Achievements Classic 1.00. Extensions should rely only on documented public fields and methods. Internal files and underscore-prefixed methods may change without a public API compatibility guarantee.