Elder Scrolls Online — Login Server

Partial Implementation Specification

Zane van Iperen

Revision	Date	Commit
v1.0	2026-08-29	be7f883a3c354d00f4b5b5304749f7f545cf3e18

This is a **partial** specification, derived from captured client/server traffic (clients 7.0.6, 2021 and 8.3.8, 2023). The login flow has not been verified against the live service since 2023, and the production protocol may have changed. See the disclaimer in Chapter 1.

Contents

1. Introduction

This document is a self-contained, **partial** specification of the **intended** behaviour of the *Elder Scrolls Online* (PC/Mac) login/authentication server, written to be handed to an implementer. It describes what a correct implementation must do — endpoint by endpoint — including the wire format exactly as the game client expects it.

Disclaimer

This is a **partial** implementation specification. It was produced by reverse-engineering the Elder Scrolls Online PC/Mac login protocol from captured traffic (client 7.0.6, June 2021) and from work against client 8.3.8 (Update 38 / Necrom era, 2023). The login flow has not been tested against the live service since 2023.
A direct probe of the live service on 2026-08-23 confirmed the announcement, realm-listing, and client-error endpoints still respond with the shapes documented here, but the login endpoints (`/login_queue/*`) returned HTTP 404[1] — the production login paths appear to have changed. An implementation built from this document may therefore not work against the current game client or server.
Treat this document as an archival record of the protocol as it existed in the 2021–2023 era, not as a guarantee of current behaviour. Where behaviour differs between the captured eras, this document says so: see the 2016-era notes in §11 and the client-version note in the progress endpoint (§5.2).

1.1. Purpose and Scope

This server implements the **login/authentication** stage of *The Elder Scrolls Online* for PC/Mac clients. It is **not** a game server: it does not host a world, simulate characters, or accept gameplay connections. It handles:

1. Account authentication (username/email + password).
2. The login queue / progress polling.
3. One-time-password (OTP) challenge for untrusted devices.
4. Realm listing and (logically) realm reservation.

[1] The response body read `404 page not found` — the default output of Go's `net/http` server — an indicator that the live login service is written in Go.

5. Announcements / message-of-the-day.
6. Client error collection.

The client is pointed at this server via its `Platforms.xml` (see §1.2.1).

1.2. Conventions and Terminology

1.2.1. Pointing the client at the server

The game reads `<install>/game/client/Platforms.xml`, which lists *platforms*. Each platform carries a `login_service_url` (the base of the login endpoints), a `customer_service_url` (the upstream customer-service API, **not** served by this server; copy it verbatim), and a `default_realm_id`. A real 8.3.8-era `Platforms.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<platforms>
  <platform>
    <name>Live</name>
    <login_service_url>https://live-services.elderscrollsonline.com</login_service_url>
    <customer_service_url>https://api-prod.cs.bethesda.net/prod</customer_service_url>
    <default>false</default>
    <default_realm_id>4000</default_realm_id>
  </platform>
  <platform>
    <name>Live-EU</name>
    <login_service_url>https://live-eu-services.elderscrollsonline.com</login_service_u
    ↪ rl>
    <customer_service_url>https://api-prod.cs.bethesda.net/prod</customer_service_url>
    <default>false</default>
    <default_realm_id>4001</default_realm_id>
  </platform>
</platforms>
```

The `Live` / `Live-EU` entries point at the production service. To intercept traffic, point the client at a local HTTP proxy that forwards to the upstream servers. A complete example of such a proxy, including the extended `Platforms.xml` it generates, is given in Appendix B.
To point the client at a local server instead, add a `<platform>` entry whose `login_service_url` points at it:

```
<platform>
  <name>Local</name>
  <login_service_url>http://127.0.0.1:8080</login_service_url>
  <customer_service_url>https://api-prod.cs.bethesda.net/prod</customer_service_url>
  <default>false</default>
  <default_realm_id>4002</default_realm_id>
</platform>
```

`default_realm_id` is the `realm_id` the client sends at login. Use a value the server does **not** advertise as a real realm (e.g. 4002) if you want the client to show the realm-selection screen rather than auto-connecting.

1.2.2. Terms

Term	Meaning
device ID	A 128-character uppercase-hex string identifying the client machine.
session UUID	A UUID the server issues for one login attempt; the client echoes it back on subsequent calls.
depot	The version of the client's game files. `4000` = PC/NA.
realm	A connectable game server. The client only lists realms whose `depot_id` matches its own.
OTP	One-time password required the first time a device logs in to an account.

1.2.3. Region / realm identifiers

ID	Meaning
`4000`	NA megaserver (also the PC/NA `depot_id`)
`4001`	EU megaserver

Language codes are two letters. The captured auth traffic uses `en`. The auth endpoint requires a 2-letter code (§5.1), and the announcement endpoint accepts any 2-letter code (§5.7).

2. Transport and Serialization

2.1. HTTP

- HTTP/1.1.
- **Requests:** `POST` bodies are `application/x-www-form-urlencoded`. `GET` endpoints read query parameters. (An implementation may merge query-string and form parameters, but need not; the client sends one or the other and does not rely on merging.)
- The client sends `User-Agent: eso/<major>.<minor>.<build>.<changelist> (<env>; <a>; <b> ; <platform>)`, e.g. `eso/7.0.6.2256692 (live; public; client; win)`.

2.2. Content Negotiation

Choose the response format from the request's `Accept` header:

- If the header contains `application/xml` (substring match) → XML (`application/xml`).
- Otherwise (`*/*`, `application/json`, absent, or any other value) → JSON.

Both responses carry a `charset=utf-8`. The client always asks for XML (`Accept: application/xml`), but the JSON path must also work.

2.3. Response Envelope

Every response is a single envelope. XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
    <response>...endpoint-specific payload...</response>
    <result_code>2000</result_code>
    <result_message>success</result_message>
</zos_platform_response>
```

JSON wraps the same object under one key:

```
{
  "zos_platform_response": {
    "response": ...,
    "result_code":2000,
    "result_message":"success"
  }
}
```

- `result_code` is an integer (see §3).
- `result_message` is a human-readable string (see §3 for canonical text).
- Omit `response` when there is no payload. (Exception: the client-error endpoint returns an empty string payload, serialised as `<response></response>` / `"response":""`.)

2.4. Serialization Rules (normative)

Serialization is strict. A correct implementation must:

1. **Keep newlines literal.** Inside `<data>` and `<message>` text, emit real newline characters — never `
`.
2. **Keep message bodies as raw inner XML.** `<data>...</data>` (announcements) and `<message>...</message>` must contain the text verbatim (newlines and `"` preserved), not escaped text.
3. **Emit `error_body` as embedded JSON.** In the auth-failure error block, `<error_body>` contains a JSON object serialised as an escaped string:

```
<error_body>{&#34;password&#34;:&#34;incorrect&#34;}</error_body>
```

4. **Serialize booleans as lowercase `true`/`false`**, and integer flags (e.g. `trusted_machine`, `streaming_status`, realm `*_allowed`/`*_match` fields) as `1`/`0`.
5. **Serialize timestamps** `ctime`/`expires_after_unix_time` as integer Unix seconds; `eta.time_last` as a float (seconds).[2]
6. **Use camelCase for** `reservation_result.connectPort` and `connectAddress` (all other fields are snake_case).
7. **Field order is not significant.** The client reads elements by name, so sibling elements may appear in any order. The examples use a consistent order only for readability and element-order parity with the captures; a standard XML serializer (struct-, map-, or DOM-based) is sufficient — no order-preserving serializer is required.

[2] The captured `time_last` values render as e.g. `1.68363982e+09` — the float format Go's `strconv` produces — another indicator that the live login service is written in Go.

8. **Begin every XML response with** `<?xml version="1.0" encoding="UTF-8"?>` (shown in the golden test vectors, §10).

9. **Mirror the XML tree in JSON.** The JSON object mirrors the XML element tree under `zos_platform_response`: scalar elements become JSON scalars (numbers stay numbers), list-typed elements (e.g. `<entitlements>`) become arrays — even when the XML carries a single item — and empty elements become `""` (observed for scalars; the same rule is applied to empty containers such as `<state_data></state_data>`), and `<data>`/`<message>` text becomes a JSON string with newlines escaped as `\n`. `error_body` remains a JSON **string** whose value is the embedded JSON object text (double-encoded), matching the XML escaped form.

3. Result Codes

Every response envelope carries a `result_code`. The canonical codes and messages are:

Code	Canonical `result_message`	Meaning
2000	`success`	Success
3000	`maintenance`	Server in maintenance (HTTP 503)
5000	`Missing Mandatory Parameters`	A required parameter was absent (observed message: `Missing Mandatory Parameters: <name>`)
5001	`Invalid Parameter Value`	A parameter failed to parse/validate
5002	`Path cannot be found`	Unknown path
5003	`Invalid Session ID`	Bad/unknown session UUID
5004	`Invalid Input Received`	Reserved
5008	`Client version not supported`	Client too old (HTTP 409)
6001	`Invalid request method for endpoint`	Wrong HTTP method
6009	`Access not allowed`	Reserved
6100	`endpoint is deprecated`	Endpoint removed (HTTP 503)
8000	`Invalid email/passwd`	Bad credentials (inside auth error block)
11000	`Invalid queue state`	Invalid state for the requested operation
9999	`Unknown error`	Unhandled error

`result_message` need not equal the canonical string — the client reads `result_code`, not the text. Of the canonical strings, 2000, 3000, 5000, 5008, 8000 and 11000 are reproduced in the captures accompanying this document and 6100 in a live probe; the remainder come from analysis of live responses during reverse-engineering and are not reproduced here (the captured 5001 messages are `no session Found`, `Invalid Parameter Value: invalid Device ID`, and `Invalid Parameter Value: invalid UUID length: N`, and 5003 `Invalid UUID received: uuid=...`).

3.1. Client-visible `status` Codes

The `POST /login_queue/progress` response carries an integer `status` that drives the client's UI/state machine:

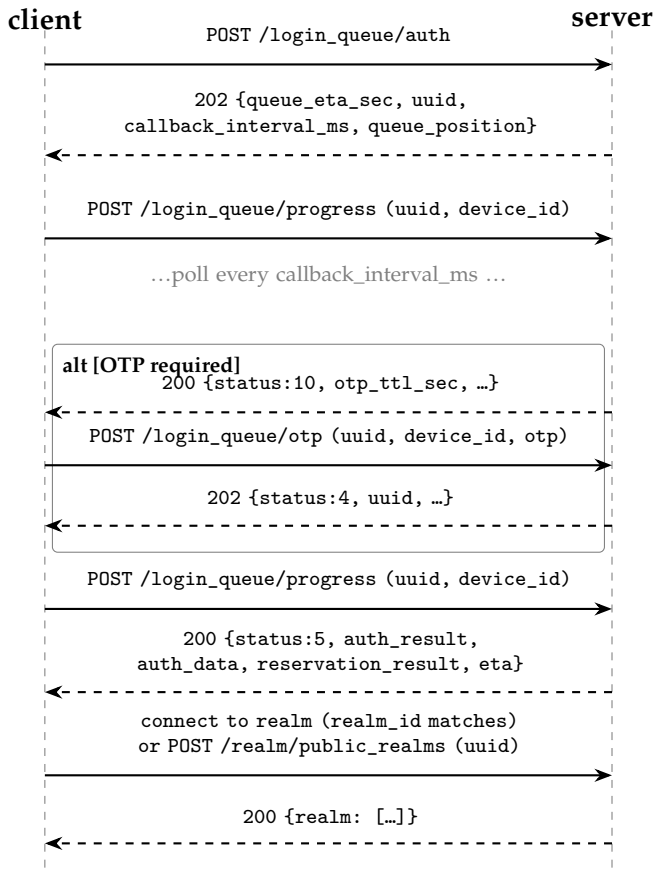
`status`	Meaning
0	Error occurred.
1	Unknown session UUID — client re-polls with a longer interval.
2	Go to `/realm/public_realms` (realm-selection screen).
3	Login information was incorrect.
4	In login queue / reservation waiting — keep polling.
5	Successful auth — auto-connect (if realm matches).
6	Reservation failed.
7	Reserved; treated by the client as `status` 1.
8	Some error.
9	Some error.
10	OTP required.
11	Cancels the login box.
12	Too many invalid logins (account locked).
13	Reserved; treated by the client as `status` 2.

A correct implementation emits only **1, 3, 4, 5, 6, 10** in normal operation; the remaining codes are reserved and not emitted. The meanings given for 0, 2, 7–9 and 11–13 are client-side: the server never emits them, and they were determined from client-side behaviour during reverse-engineering.

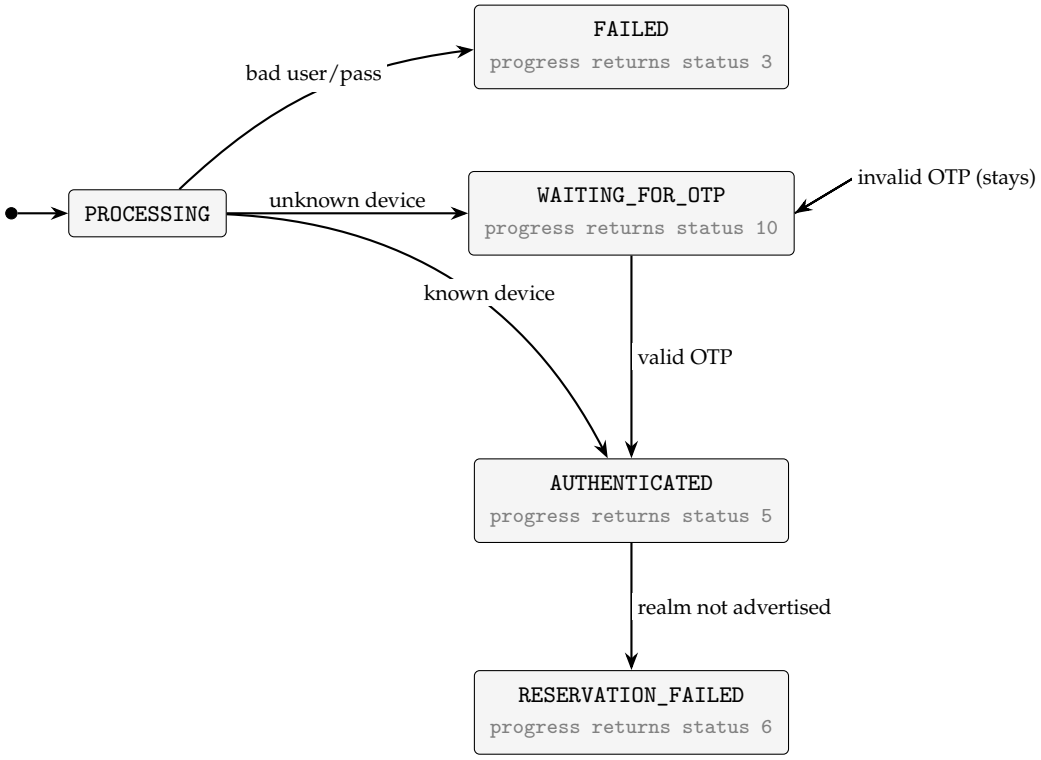
4. The Login Flow

4.1. Sequence

The full client-driven sequence:



4.2. Server-side Session State Machine



This state model is illustrative: it is a convenient way to reason about the login flow, not an observation of the live server's internals (the real server's state machine, if it has one, is not visible from the wire and is presumably far larger). What is wire-observable are the display strings below, which appear as envelope `result_messages` and inside 11000 queue-state messages (an additional wire-observed display string, `InvalidState`, appears in a captured 11000 message — §5.5).

The authentication work is completed before the `/login_queue/auth` 202 is returned. The auth endpoint returns 202 immediately (unless short-circuited, §5.10) with the session UUID; the first `progress` poll then returns the session state (`status` 3 / 10 / 5 / 6). A session still in `PROCESSING` (only reachable after a crash during authentication) answers a `progress` poll with `status` 4 (keep polling).

State names and their display strings, used as the `result_message` of the state's terminal or OTP-required response and interpolated into 11000 queue-state messages (the `PROCESSING` state's `progress` response instead uses `Reservation Waiting`, below):

State	Display string	Terminal?
PROCESSING	Authentication Pending	no
WAITING_FOR_OTP	OTP Waiting	no
AUTHENTICATED	Reservation Successful	yes
FAILED	Authentication Failed	yes
RESERVATION_FAILED	Reservation Failed	yes

PROCESSING never appears in a progress response (authentication is synchronous): the auth endpoint's 202 uses Authentication Pending, and a progress poll against a PROCESSING session returns status 4 with result_message Reservation Waiting (§5.2). POST /login_queue/cancel removes the session only for the non-terminal states (PROCESSING, WAITING_FOR_OTP); cancelling a terminal state returns 11000.

4.3. Terminal Success (status 5) and Failed Reservation (status 6)

On success return the full success payload: auth_result + auth_data + reservation_result + eta (see §5.2). The reservation_result is how the client learns the game server's address/port to connect to.
status 5 (auto-connect) is returned when the submitted realm_id matches an advertised realm (§7). When it does not, the reservation fails and status 6 (Reservation Failed) is returned instead (§5.2). The server never emits status 2 (it never appears in the captures): the client reaches the realm-selection screen on its own when the requested realm_id is not advertised, and then lists realms via POST /realm/public_realms.

5. Endpoint Reference

All form parameters are strings; `0/1` denote booleans.

5.1. `POST /login_queue/auth`

Begin authentication.
Request (form-encoded):

Field	Required	Notes
`email_address`	yes	Username **or** email (how the server interprets it is an implementation choice, §6.1).
`password`	yes	Plaintext (how the server verifies it is an implementation choice, §6.2).
`realm_id`	yes	e.g. `4000`.
`device_id`	yes	Exactly 128 hex characters (`[0-9A-F]{128}`).
`trusted_machine`	yes	`0/1`.
`language`	yes	2-letter code.
`hardware_type`	yes	integer (`1`).
`streaming_status`	yes	`0/1`.

Response: HTTP 202 Accepted, `result_code` 2000, `result_message` `Authentication Pending`:

Field	Type	Notes
`queue_eta_sec`	float	Non-negative queue wait; `0` is acceptable.
`uuid`	UUID	The session UUID to use in subsequent calls.
`callback_interval_ms`	int	`2000` (normative).
`queue_position`	int	`1` (normative).

Errors: missing param → 400/`5000`; invalid `device_id` (not 128 hex characters) → 400/`5001`; invalid `language` (not a 2-letter code) → 400/`5001`; invalid `trusted_machine` or `streaming_status` (not `0` or `1`), non-integer `hardware_type`, or non-numeric `realm_id` → 400/`5001`. Observed instances: a missing `logmessage` → 5000 on the client-error endpoint (§5.9), and an invalid `device_id` → 5001 (`Invalid Parameter Value: invalid Device ID`) on the live service; the remaining rules follow the generic invalid → 5001 mapping (§9).

5.2. POST /login_queue/progress

Poll the session state.

Request: `uuid` (required), `device_id` (required; checked against the session's device). A malformed `uuid` → 400/`5001` (observed message: `Invalid Parameter Value: invalid UUID length: N`; a missing `uuid` fails the same parse, reporting length 0).[3] A present-but-mismatched `device_id` is treated as an unknown session (`status` 1); a progress poll without `device_id` was not observed in the captures.

Response: HTTP 200, `result_code` 2000. The `response` object varies by state.

5.2.1. Unknown session (`status` 1)

Envelope `result_message` `UUID Not Found`. A `device_id` that does not match the session is treated the same as an unknown session:

Field	Value
`status`	1
`state_data`	empty element
`otp_ttl_sec`	0
`queue_eta_sec`	0
`callback_interval_ms`	10000
`queue_position`	0

5.2.2. Bad credentials (`status` 3)

Envelope `result_message` `Authentication Failed`, plus an error block (§5.2.7) and a minimal `auth_result`/`auth_data` (field set as shown in §10.4).

All captured bad-credential responses (client 7.0.6 era) show `status` 3. On the 8.3.8-era service, a failed login was instead observed to send the client to the OTP prompt (on the live service; not reproduced in the captures): the session is treated as OTP-required and `progress` answers `status` 10 (`OTP Waiting`) again and again (10 → 10 → …) until the attempt is abandoned.

5.2.3. OTP required (`status` 10)

Envelope `result_message` `OTP Waiting`:

[3] `invalid UUID length: N` is the `Parse` error of Go's `google/uuid` package (`https://github.com/google/uuid`); another indicator that the live login service is written in Go.

Field	Value
status	10
state_data	empty element
otp_ttl_sec	86400
queue_eta_sec	0
callback_interval_ms	1000
queue_position	0

The queue fields (otp_ttl_sec, queue_eta_sec, callback_interval_ms, queue_position) appear only in progress responses with status 1, status 4, and status 10; of these, otp_ttl_sec appears only in status 1 and status 10.

5.2.4. Still processing (status 4)

Only reachable after a crash mid-auth; no progress capture shows this state (the OTP endpoint's 202 carries the same status 4 with queue_position 1, §5.3), so the values below are design choices. Envelope result_message Reservation Waiting:

Field	Value
status	4
state_data	empty element
queue_eta_sec	0
callback_interval_ms	2000
queue_position	0

5.2.5. Success (status 5)

Envelope result_message Reservation Successful, with full state_data.data:

```
status: 5
state_data.data:
  auth_result:
    uuid                  (session UUID)
    access_flags          0
    entitlements_mask     int
    entitlements          [int, ...]      (list)
    account_name          string          (stored username - even when login was by
    ↪  email)
    country               string          (account's country; "US"/"AU" in the captures)
    user_id               int
    email                 string          (account email)
    sg_new_ip             bool
    console_id            0
    console_subscription  bool
    is_transfer           bool
    master_account_id     int
```

```
    game_account_type_ids 23
    platform              "pc"
  auth_data:
    email_address         (echoes the submitted login string verbatim - not the account
    ↪  email)
    language              (as submitted)
    realm_id              (as submitted)
    ps4_auth_code         ""
    hardware_type         (as submitted)
    streaming_status      (as submitted)
    psn_client_region     ""
  reservation_result:
    uuid                  (session UUID)
    succeeded             true
    realm_name            "NA Megaserver"
    connectPort           int    (game server port - camelCase!)
    connectAddress        string (game server address - camelCase!)
    realm_id              (as submitted)
    depot_id              4000
    retry                 false
  eta:
    average_wait          0
    queue_size            0
    time_last             float (current Unix time, seconds)
```

`country`, `entitlements_mask`, `entitlements`, `master_account_id`, `game_account_type_ids`, `sg_new_ip`, and `platform` have no natural source in the stored user record; a minimal implementation uses defaults (`country:US`, `entitlements_mask:0`, `entitlements:[]`, `master_account_id:user_id`, `game_account_type_ids:23`, `sg_new_ip:false`, `platform:pc`). `connectAddress`/`connectPort` come from the matched realm's configuration (§7). The repeated `entitlements` elements reflect the account's DLC entitlements and are omitted when it has none (observed: no elements with `entitlements_mask` 0).

5.2.6. Reservation failed (`status 6`)

Returned when the submitted `realm_id` is not an advertised realm. Envelope `result_message` `Reservation Failed`. The full `auth_result`, `auth_data`, and `eta` are still present; the `reservation_result` has `succeeded = false`, empty `realm_name`, `connectPort = 0`, `realm_id = 0`, `depot_id = 0`, `retry = false`, and **omits** the `uuid` and `connectAddress` fields entirely.
Once a terminal response (`status` 3/5/6) has been delivered, subsequent `progress` polls on the same session return the same terminal payload (idempotent).

5.2.7. The auth-error block

```
error:
  http_code        401
  result_code      8000
  result_message   "Invalid email/passwd"
  error_body       '{"password":"incorrect"}'   (JSON string, escaped in XML)
```

In JSON, error_body remains a **string** whose value is the same embedded JSON object text ("password":"incorrect" double-encoded), mirroring the XML escaped form.

5.3. POST /login_queue/otp

Submit the OTP for an untrusted device.
Request: uuid, device_id, otp (all required).
Response: HTTP 202 Accepted, result_code 2000, result_message Reservation Waiting:

Field	Value
uuid	a throwaway reservation UUID — distinct from the session UUID (the client keeps polling with the session UUID issued by POST /login_queue/auth); how the server generates or persists it is an implementation choice
queue_eta_sec	float; non-negative queue wait, 0 acceptable
callback_interval_ms	2000
queue_position	1
status	4

On success, record the device as trusted for the account. On an invalid OTP, keep the session in the OTP-required state, do **not** record the device, and respond HTTP 200 with the status 10 (OTP Waiting) payload again (§5.2). Retries are unlimited. An OTP submitted against an unknown UUID or with a mismatched device_id → 5003; against a session not awaiting OTP → 11000 (Invalid queue state <state display string> for otp call: <uuid>; the call-type suffix is an implementation choice, §5.5); both errors carry HTTP 200, as with cancel (§5.4).

5.4. POST /login_queue/cancel

Cancel a pending login.
Request: uuid, device_id (both required; missing → 400/5000).
Behaviour: if the session exists and both uuid and device_id match, remove it and respond **HTTP 200** with result_code 2000, result_message success, and an empty envelope. A device_id that does not match is treated as an unknown UUID (5003). A cancel against a terminal session (FAILED, AUTHENTICATED, or RESERVATION_FAILED) yields result_code 11000 (Invalid queue state <display string> for cancel call: <uuid>, where <display string> is from the state table, §4.2); an unknown UUID yields 5003 (Invalid UUID received: uuid=...). All cases return HTTP 200 — the error is carried in the envelope.

5.5. POST /login_queue/reservation

Request: uuid, device_id, realm_id (all required).

Reserve a realm slot. This endpoint is **not part of the normal login flow** — the reservation is already delivered via `status` 5's `reservation_result`. A minimal implementation returns `result_code` 11000 (`Invalid queue state InvalidState for reservation call: <uuid>`; the call-type suffix is an implementation choice — captured messages read `... for cancel call:` and `... for progress call:`; `InvalidState` is the display string captured in `reservation_1.xml`) with HTTP 200, and `result_code` 5003 (`Invalid UUID received: uuid=...`) with HTTP 404 for an unknown UUID.

5.6. POST /realm/public_realms

List realms.
Request: `uuid` (session UUID). Any existing, unexpired session qualifies, regardless of its state. A missing `uuid` → 400/5000; an unknown or expired UUID → HTTP 400/5001 (`no session Found`).
Response: HTTP 200, `result_code` 2000, `result_message` `Public Realm Listing`, `response` = every realm whose `depot_id` is `4000` (the only supported PC depot) — fields in §7. XML serialises each realm as a `<realm>` element; JSON serialises them as a `realm` array. If no realm matches, return an empty list.

5.7. GET /announcement/announcement

Query: `message_type` (repeatable; values `announcement`, `motd`, `server_alert`), `language` (2-letter code). Both required (missing → 400/5000). An unrecognised `message_type` value is ignored. `language` is any 2-letter code; an unsupported language simply returns no matches (no validation error). A capture (`captures/motd.json`) shows the response to a request for all three types at once (the request line below was verified against a client request recorded during reverse-engineering):

```
GET /announcement/announcement?message_type=server_alert&message_type=announcement&mess⌋
↪  age_type=motd&language=en
```

Response: HTTP 200, `result_code` 2000, `result_message` `success`. `response` contains one entry per announcement, each `{ data, message_type, language }`:

- XML: one `<response>` element **per announcement** (repeated), each containing `<data>`, `<message_type>`, `<language>`.
- JSON: `response` is an array of `{data, message_type, language}` objects.

Return announcements matching the requested language *and* any of the requested types. `data` carries the message text (raw, newlines preserved). Return an empty list when nothing matches.

5.8. GET /announcement/message (deprecated)

Deprecated; not used by the login flow. This endpoint is optional to implement — the production server responds HTTP **503** with `result_code` 6100 (`endpoint is deprecated`). If implemented:
Query: `announcer_id` (integer; missing → 400/5000). **Response: HTTP 200**, `result_code` 2000, `result_message` `success`, `response` = the message(s) for that announcer: XML uses one `<response><message>...</message></response>` per entry; JSON uses an array of `{"message": ...}` objects. Return an empty list for an unknown `announcer_id`; a NULL message serialises as an empty string.

5.9. GET /log/collect_esoclienterror

Query: `logmessage` (required) — dotted numeric, e.g. `301.3.1.816.10061` (corresponding to the client's `Error <n> (<a>:<b>:<c>:<d>)` strings). A missing `logmessage` → 400/5000.
Response: HTTP 200, `result_code` 2000, `result_message` `success`, `response` = empty string (`<response></response>` / `"response":""`). Persist the message verbatim (no format validation required); an implementation may store it with the remote address, user-agent, and timestamp, but the wire response does not depend on it.

5.10. Maintenance and out-of-date-client

Two special conditions short-circuit the login endpoints (§5.1–§5.5):

- **Maintenance:** (may be activated by an operator-controlled flag) HTTP **503**, `result_code` 3000, `result_message` `maintenance`, with a `response` carrying the retry hint (the client disables login for `ttl_seconds`):

```
{
  "message": "Services are currently unavailable.",
  "ttl_seconds": 300
}
```

- **Out-of-date client:** HTTP **409**, `result_code` 5008, `result_message` `Client version not supported`, with **no** `response` payload (the 2016-era 5008 carried a `message`/`ttl_seconds` payload, §11). Detect an out-of-date client by parsing `User-Agent` (`eso/<major>.<minor>.<build>.<changelist>`) and rejecting it when the version is below a configured minimum (e.g. `7.0.0.0`).

The version gate applies to the login endpoints (§5.1–§5.5). A request with an unparseable `User-Agent` was observed to be rejected: a non-client request in 2023 received the HTTP 409/5008 response. When both conditions apply, the out-of-date check wins (HTTP 409/5008).

6. Authentication Backend Behaviour

6.1. User lookup

The `email_address` field carries a single login string; how the server interprets it is an implementation choice, not part of the protocol. One common approach:

- If the login contains `@`, treat it as an **email** (normalise: trim leading/trailing whitespace from the whole input; lowercase only the domain — the part after the last `@`; preserve the local part exactly, including case). Otherwise treat it as a **username** (exact, case-sensitive match).

The protocol only requires that a valid login resolves to an account; an unknown account → the session is `FAILED` and `progress` returns `status` 3.

6.2. Password verification

The `password` field is submitted in plaintext (form-encoded); how the server verifies it is an implementation choice. One common approach:

- Verify with **bcrypt** (cost 11; accept `$2a$`, `$2y$`, and `$2b$` prefixes).

A mismatch → `FAILED` → `status` 3.
These `status` 3 outcomes describe the 7.0.6-era service; on the 8.3.8-era service failed logins were instead observed to route to the OTP prompt (§5.2).

6.3. Device trust and OTP

- A device is trusted iff its `device_id` is recorded against the account.
- On login, if the device is **not** trusted → `WAITING_FOR_OTP` (status 10). If it is trusted → `AUTHENTICATED` (progress `status` 5, or `status` 6 if the requested realm is not advertised).
- On successful OTP submission, **persist** the device, then mark the session authenticated.
- `trusted_machine` is informational only; do not use it to skip OTP.

OTP mechanism: the wire protocol does not mandate any particular OTP scheme. The server must be able to verify the `otp` value the client submits against a per-user secret provisioned out-of-band (there is no self-service enrollment endpoint).
Implementor's note: how the OTP reaches the user is an implementation choice; the wire protocol only requires that the server can verify the submitted `otp`. Examples:

- Print a one-time code to the server console for the operator to type into the client.
- RFC-6238 TOTP (SHA1, 6 digits, 30-second period, per-user base32 secret, counter `floor(unix_seconds / 30)`, accepting the current and adjacent steps (±1) for clock skew) — convenient because the per-user secret can be loaded into an authenticator app.

The 2021-era live service emailed the code to the account's registered address (observed); like the examples above, that is only a suggestion. A user with no stored secret cannot be OTP-challenged; how to handle that (e.g. treat the device as trusted, or reject the login) is an operator choice. Any scheme the server can verify works.

6.4. Sessions

- A session is identified by its UUID and tracks: `device_id`, login name, `realm_id`, `language`, `hardware_type`, `streaming_status`, state, and (once known) the user id/email/username.
- Whether sessions survive a server restart is an implementation choice; if they do not, in-flight login attempts are interrupted (the client re-polls and eventually gets `status` 1).
- Sessions expire after a TTL (recommended default 300 seconds, configurable); an expired session behaves like an unknown UUID (`status` 1). This applies to terminal sessions too: terminal idempotency (§5.2) holds only until the session expires. Sessions in `WAITING_FOR_OTP` must survive at least as long as the advertised `otp_ttl_sec` (86400): use a longer TTL for that state or exempt it from expiry.
- `progress` must check the `device_id` matches the session's — an invalid session or mismatched device is treated as an unknown session (`status` 1) and must not leak another session's state.

A server exposes no admin or provisioning endpoints: users (including their OTP secret), announcements, and realms are provisioned out-of-band by the operator.

7. Realms

A realm advertises a connectable game server. The client shows a realm only if its `depot_id` matches the client's game depot; if its `realm_id` matches the one supplied at login, the client auto-connects.
Realm object fields (exact wire names):

Table 7.1: Realm object fields

Field	Type	Notes
`require_major_match`	int-bool	
`world_type`	string	`globby`
`realm_rest_server_url`	string	usually empty
`max_population`	int	
`realm_name`	string	e.g. `NA Megaserver`
`realm_status`	int-bool	
`build_version`	int	
`build_changelist`	int	
`p4_allowed`	int-bool	
`live_update_allowed`	int-bool	
`release_allowed`	int-bool	
`minor_version`	int	
`debug_allowed`	int-bool	
`entitlements`	int (XML) / array (JSON)	multiple values serialise as repeated `<entitlements>` elements in XML
`entitlements_list`	string	usually empty (`""` in JSON)
`protocol_version`	int	
`lock_state`	int	
`require_build_match`	int-bool	
`require_minor_match`	int-bool	

Continued on next page

Table 7.1: Realm object fields (Continued)

Field	Type	Notes
expires_after_unix_time	int	Unix seconds
lobby_protocol	int	
population	int	
require_protocol_match	int-bool	
ctime	int	Unix seconds
major_version	int	
region_protocol	int	
tool_allowed	int-bool	
realm_id	int	4000 NA, 4001 EU
depot_id	int	4000 = PC client's depot

Serve at least one realm; the standard deployment serves two: NA 4000 and EU 4001. Both realms must advertise depot_id 4000 (the PC client's depot). A realm with any other depot_id is not listed by the PC client.
The realm configuration also carries a connect_address and connect_port (the game server the client connects to); status 5 fills reservation_result.connectAddress / connectPort from the matched realm. A minimal single-server deployment might use e.g. 127.0.0.1:24507 (24507 is the connectPort of the success vector, §10.5).
For the EU realm, copy the realm-list vector (§10.6) field set and change realm_id to 4001, realm_name to EU Megaserver, and connect_address/connect_port as configured; ctime/expires_after_unix_time may be fixed constants (the vector values are acceptable). The vector shows the complete realm field set; element order is not significant (§2.4).

8. Announcements

Three types: `announcement`, `motd`, `server_alert`. Store them per `(language, type)`. The message text may contain ESO markup like `|c76BCC3|H1:url:http://...|h here|h|r` — pass it through verbatim; the `|c...|`, `|H...|h...|h|r` tags are client-side rendering hints (here: coloured hyperlink text) and carry no protocol meaning.
The wire format — one `<response>` element per announcement, each `{ data, message_type, language }` — is documented in §5.7.

8.1. Selection

The client requests a set of message types and a 2-letter `language`; the server returns the stored records whose language matches and whose type is one of the requested types (§5.7). The captured requests ask for all three types at once; only `announcement` and `motd` records were ever observed — no `server_alert` record appeared. The observed announcements were served in `en`, `ru` and `jp`.

8.2. Example records

The records below are from captured traffic, era-labelled. Newlines, spacing and any ESO markup are part of the message. The 2021-era announcement was also served in Russian and Japanese; only the `en` form is shown here.

8.2.1. Announcement (client 7.0.6 era, 2021)

```
Welcome to The Elder Scrolls Online and patch 7.0.5 on the North American PC/Mac
↪  megaserver.

The Elder Scrolls Online: Blackwood and Update 30 are now live! Discover the wilds of
↪  Blackwood, recruit new Companions, and continue (or begin!) your Gates of Oblivion
↪  adventure. To get started, create a new character or travel directly to the Leyawiin
↪  Outskirts wayshrine.
Rededicate your Champion Points for free during the Champions Reforged mini-event,
↪  starting now and running until June 15.
```

8.2.2. MOTD (client 7.0.6 era, 2021)

```
NOTICE: The North American PC/Mac megaserver is currently unavailable while we perform
↪  maintenance.
```

8.2.3. Announcement (client 8.3.8 era, 2023)

```
Welcome to The Elder Scrolls Online and patch 8.3.8 on the North American PC/Mac
↪  megaserver.

Update 38 is now available for testing on the PTS! To participate, open your ESO
↪  launcher, navigate to Settings, and select "Show Public Test Environment." Have fun!

Begin your Shadow Over Morrowind saga with the Necrom Prologue quest "Eye of Fate" now
↪  available in the in-game Crown Store.

Master the Arcanist class, explore the Telvanni Peninsula and realm of Apocrypha, and
↪  defend the secrets of Hermaeus Mora in The Elder Scrolls Online: Necrom.
↪  Pre-purchase now to get access to bonus in-game rewards at launch June 5.
```

8.2.4. Announcement with markup (2016, game patch 2.3.10 era)

```
Welcome to the Elder Scrolls Online: Tamriel Unlimited on the North American PC/Mac
↪  megaserver!

The Dark Brotherhood DLC game pack and base game patch are now available to test on the
↪  PTS! Read the full patch notes
↪  |c76BCC3|H1:url:http://forums.elderscrollsonline.com/en/discussion/261799/|h
↪  here|h|r.

The Thieves Guild DLC game pack and base-game patch (2.3.9) are now available on PC/Mac!
↪  Join the Thieves Guild of Abah's Landing to become the newest recruit in their
↪  organization of pickpockets, burglars, robbers, and thieves. This latest DLC game
↪  pack is included with an active ESO Plus membership, or can be purchased from the
↪  Crown Store.
```

The `|c...|`, `|H...|h...|h|r` tags render as coloured hyperlink text in the client; the server stores and returns them unmodified.

9. Error Handling Summary

Condition	HTTP	`result_code`
Missing required parameter	400	5000
Invalid/unparseable parameter	400	5001
Unknown path	404	5002
Wrong method	405	6001
Unknown session UUID (progress)	200	2000 (`UUID Not Found`)
Unknown session UUID (cancel)	200	5003
Unknown session UUID (reservation)	404	5003
No session on `public_realms`	400	5001
Bad credentials	200 (progress)	2000 + inner 8000
Invalid OTP	200	2000 (`OTP Waiting`)
OTP for unknown UUID	200	5003
OTP for session not awaiting OTP	200	11000
Reservation invalid state	200	11000
Maintenance	503	3000
Out-of-date client	409	5008
Deprecated endpoint	503	6100
Unhandled	500	9999

The response bodies and message strings for each case are given in the endpoint reference (§5) and the result-code table (§3). For `progress` failures the HTTP status stays **200** — the error is carried in the body (`status` + `error` block), because the client reads the body to decide state.

10. Golden Test Vectors

Complete response bodies for the key flows. Account names are redacted; session UUIDs are illustrative values. These are normative examples: a correct implementation must produce the same structure and static field values (status/result codes, message strings, field sets, booleans, flags). Server-generated values — the session `uuid`, `queue_eta_sec`, and `time_last` — are illustrative and vary per request. The vectors describe the 2021-era (client 7.0.6) shapes; the older 2016-era records use a different field set (§11).

10.1. Pending Auth (`POST /login_queue/auth`) — HTTP 202

```xml
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <queue_eta_sec>0.009561846</queue_eta_sec>
    <uuid>2de06c14-4166-4f66-b3ae-b97d35ed9739</uuid>
    <callback_interval_ms>2000</callback_interval_ms>
    <queue_position>1</queue_position>
  </response>
  <result_code>2000</result_code>
  <result_message>Authentication Pending</result_message>
</zos_platform_response>
```

The corresponding capture is reproduced in §A.3.2.

10.2. OTP Required (`progress`, `status` 10) — HTTP 200

```xml
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <status>10</status>
    <state_data></state_data>
    <otp_ttl_sec>86400</otp_ttl_sec>
    <queue_eta_sec>0</queue_eta_sec>
    <callback_interval_ms>1000</callback_interval_ms>
    <queue_position>0</queue_position>
  </response>
  <result_code>2000</result_code>
  <result_message>OTP Waiting</result_message>
```

```
</zos_platform_response>
```

The corresponding capture is reproduced in §A.2.

10.3. OTP Submit (`POST /login_queue/otp`) — HTTP 202

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <uuid>a50fd878-a253-4331-8677-2e3c90622584</uuid>
    <queue_eta_sec>0.01327475</queue_eta_sec>
    <callback_interval_ms>2000</callback_interval_ms>
    <queue_position>1</queue_position>
    <status>4</status>
  </response>
  <result_code>2000</result_code>
  <result_message>Reservation Waiting</result_message>
</zos_platform_response>
```

The corresponding capture is reproduced in §A.3.5.

10.4. Bad Credentials (`progress`, `status 3`) — HTTP 200

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <status>3</status>
    <state_data>
      <data>
        <error>
          <http_code>401</http_code>
          <result_code>8000</result_code>
          <result_message>Invalid email/passwd</result_message>
          <error_body>{&#34;password&#34;:&#34;incorrect&#34;}</error_b
          ↪ ody>
        </error>
        <auth_result>
          <uuid>be848409-a654-4a22-b914-ed333f6e40c5</uuid>
          <access_flags>0</access_flags>
          <entitlements_mask>0</entitlements_mask>
          <user_id>0</user_id>
          <sg_new_ip>false</sg_new_ip>
          <console_id>0</console_id>
          <console_subscription>false</console_subscription>
```

```
          <is_transfer>false</is_transfer>
          <platform>pc</platform>
        </auth_result>
        <auth_data>
          <email_address>MyUsername</email_address>
          <language>en</language>
          <realm_id>4002</realm_id>
          <ps4_auth_code></ps4_auth_code>
          <hardware_type>1</hardware_type>
          <streaming_status>1</streaming_status>
          <psn_client_region></psn_client_region>
        </auth_data>
      </data>
    </state_data>
  </response>
  <result_code>2000</result_code>
  <result_message>Authentication Failed</result_message>
</zos_platform_response>
```

The corresponding capture is reproduced in §A.3.6.
JSON form of the same response:

```
{
  "zos_platform_response": {
    "response": {
      "status": 3,
      "state_data": {
        "data": {
          "error": {
            "http_code": 401,
            "result_code": 8000,
            "result_message": "Invalid email/passwd",
            "error_body": "{\"password\":\"incorrect\"}"
          },
          "auth_result": {
            "uuid": "be848409-a654-4a22-b914-ed333f6e40c5",
            "access_flags": 0,
            "entitlements_mask": 0,
            "user_id": 0,
            "sg_new_ip": false,
            "console_id": 0,
            "console_subscription": false,
            "is_transfer": false,
            "platform": "pc"
          },
```

```
            "auth_data": {
              "email_address": "MyUsername",
              "language": "en",
              "realm_id": 4002,
              "ps4_auth_code": "",
              "hardware_type": 1,
              "streaming_status": 1,
              "psn_client_region": ""
            }
          }
        }
      },
      "result_code": 2000,
      "result_message": "Authentication Failed"
    }
}
```

10.5. Success (`progress`, `status` 5) — HTTP 200

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <status>5</status>
    <state_data>
      <data>
        <auth_result>
          <uuid>032ef38a-d35d-11eb-8b87-d7f72a84b0bb</uuid>
          <access_flags>0</access_flags>
          <entitlements_mask>154665472</entitlements_mask>
          <entitlements>10</entitlements>
          <entitlements>22</entitlements>
          <entitlements>21</entitlements>
          <entitlements>28</entitlements>
          <entitlements>25</entitlements>
          <entitlements>20</entitlements>
          <account_name>MyUsername</account_name>
          <country>US</country>
          <user_id>22222222</user_id>
          <email>eso@example.com</email>
          <sg_new_ip>true</sg_new_ip>
          <console_id>0</console_id>
          <console_subscription>false</console_subscription>
          <is_transfer>false</is_transfer>
          <master_account_id>11111111</master_account_id>
```

```
          <game_account_type_ids>23</game_account_type_ids>
          <platform>pc</platform>
        </auth_result>
        <auth_data>
          <email_address>MyUsername</email_address>
          <language>en</language>
          <realm_id>4000</realm_id>
          <ps4_auth_code></ps4_auth_code>
          <hardware_type>1</hardware_type>
          <streaming_status>1</streaming_status>
          <psn_client_region></psn_client_region>
        </auth_data>
        <reservation_result>
          <uuid>032ef38a-d35d-11eb-8b87-d7f72a84b0bb</uuid>
          <succeeded>true</succeeded>
          <realm_name>NA Megaserver</realm_name>
          <connectPort>24507</connectPort>
          <connectAddress>198.20.200.162</connectAddress>
          <realm_id>4000</realm_id>
          <depot_id>4000</depot_id>
          <retry>false</retry>
        </reservation_result>
        <eta>
          <average_wait>0</average_wait>
          <queue_size>0</queue_size>
          <time_last>1.623774182e+09</time_last>
        </eta>
      </data>
    </state_data>
  </response>
  <result_code>2000</result_code>
  <result_message>Reservation Successful</result_message>
</zos_platform_response>
```

The corresponding capture is reproduced in §A.3.1.

10.6. Realm List (POST /realm/public_realms) — HTTP 200

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <realm>
      <build_changelist>2256692</build_changelist>
      <build_version>6</build_version>
```

```
        <ctime>1623923349</ctime>
        <debug_allowed>1</debug_allowed>
        <depot_id>4000</depot_id>
        <entitlements>0</entitlements>
        <entitlements_list></entitlements_list>
        <expires_after_unix_time>1624355349</expires_after_unix_time>
        <live_update_allowed>1</live_update_allowed>
        <lobby_protocol>0</lobby_protocol>
        <lock_state>0</lock_state>
        <major_version>7</major_version>
        <max_population>0</max_population>
        <minor_version>0</minor_version>
        <p4_allowed>1</p4_allowed>
        <population>0</population>
        <protocol_version>6</protocol_version>
        <realm_id>4000</realm_id>
        <realm_name>NA Megaserver</realm_name>
        <realm_rest_server_url></realm_rest_server_url>
        <realm_status>1</realm_status>
        <region_protocol>0</region_protocol>
        <release_allowed>1</release_allowed>
        <require_build_match>0</require_build_match>
        <require_major_match>0</require_major_match>
        <require_minor_match>0</require_minor_match>
        <require_protocol_match>0</require_protocol_match>
        <tool_allowed>1</tool_allowed>
        <world_type>globby</world_type>
      </realm>
    </response>
    <result_code>2000</result_code>
    <result_message>Public Realm Listing</result_message>
</zos_platform_response>
```

The corresponding capture is reproduced in §A.4.

11. Captured Traffic (non-normative reference)

Request/response captures from the PC/Mac login service, provided as wire-format samples only — they are **non-normative**. XML is prettified for readability; some session UUIDs, device IDs, account names and email addresses are redacted.
Direct probes of the live service confirmed the announcement, realm-listing, and client-error endpoints still produce these shapes; the legacy `/announcement/message` endpoint now returns `result_code` 6100 (`endpoint is deprecated`).
The full contents are reproduced in Appendix A.

11.1. Full login transcript (client 7.0.6, 2021)

A complete request/response transcript of two logins: a US success (`status` 5), and an EU OTP-required flow with an invalid realm (`status` 10 → OTP submit → `status` 6). The OTP value is redacted as `ABCDEFGH`.

11.2. Auth progress

- Successful login — `status` 5 (sanitised capture data, from a different capture than the vector; structurally equivalent to the success vector, §10.5).
- Login accepted — `POST /login_queue/auth` → 202 pending.
- Login accepted (8.3.8 era, 2023) — `POST /login_queue/auth` → 202 pending (same shape as the 7.0.6-era capture).
- Successful login (8.3.8 era, 2023) — `status` 5 with the same field set; the repeated `entitlements` elements are absent because the account holds no DLC entitlements (`entitlements_mask` 0).
- Invalid password — bad credentials, `status` 3.
- OTP valid — `POST /login_queue/otp` → 202 reservation-waiting.

11.3. Validation errors (live service, 2023)

- Invalid `device_id` — HTTP 400/`5001` `Invalid Parameter Value: invalid Device ID` (JSON response).

- Malformed `uuid` on progress — HTTP 400/`5001` `Invalid Parameter Value: invalid UUID length: N`.

11.4. Cancel / reservation

- Bad cancel — cancel after completion → 11000.
- Invalid cancel — cancel with an unknown UUID → 5003.
- Invalid reservation — invalid-queue-state 11000 (the captured message reads `... for progress call`).

11.5. Realm listings

- Public realm listing (NA) — XML.
- Public realm listing (NA) — JSON (`realm` and `entitlements` are arrays).
- Public realm listing (EU) — XML.
- Public realm listing with no session → 5001 `"no session Found"`.

11.6. Announcements

- Multiple announcements — JSON, requested with all three message types at once.
- Multiple announcements — XML (`announcement` + `motd`).

11.7. Client-error collection

- Error submit — 200 with an empty-string `response` (XML and JSON).
- Error submit without the `logmessage` parameter → 5000 (XML and JSON).

11.8. Maintenance / out-of-date / bad session

- Login maintenance — 503 / 3000.
- Login with a bad client version — 409 / 5008 (XML and JSON).
- Login with a bad session ID — UUID Not Found (`status` 1, `callback_interval_ms` 10000; shape re-verified against the live service in 2023).

11.9. Older-client captures (2016, game patch 2.3.10)

These are historical 2016 capture records and do **not** describe this server's behaviour; no client version is recorded in them. The 2.3.10 label follows the realm listing's `major_version` 2 / `minor_version` 3 / `build_version` 10 — the same realm-version-to-client mapping that identifies the 2021-era records (realm `major_version` 7, client 7.0.6). The era's announcement copy names the previous patch, 2.3.9 (the Thieves Guild update); 2.3.10 was a patch release with no announcement of its own. They use a different field set from the endpoint reference (§5) and golden vectors (§10); the wire-format differences from the modern (2021+) records are:

- Booleans serialise as `True/False` (modern: `true/false`).
- A failed login nests `<error_body><password>...</password></error_body>` with `http_code` 404 (modern: an escaped JSON string with `http_code` 401).
- The realm listing omits `build_changelist` and `entitlements_list`.
- The XML declaration is lowercase `utf-8` (modern: `UTF-8`).
- The OTP submit response echoes the session `uuid`, omits `status`, and reports `queue_position` 0 — the modern endpoint returns a throwaway UUID with `status` 4 and `queue_position` 1 (§5.3).
- The captured auth 202 (a new-computer login) answers `OTP Waiting`, where the modern-era 202 answers `Authentication Pending` in both its trusted and OTP-required flows.
- An out-of-date-client (409/5008) response carries a `message` + `ttl_seconds` payload (the modern 5008 has none).
- `auth_data` carries only `realm_id`, `email_address`, `ps4_auth_code` and `language` — no `hardware_type`, `streaming_status` or `psn_client_region`.
- `auth_result` omits `platform` and the repeated `entitlements` list, and instead carries empty `legal_docs`, `console_entitlements` and `external_name` elements.
- `reservation_result.depot_id` is a boolean (`False`), not the numeric depot id of the modern era.
- The bad-credentials (`status` 3) payload includes an `eta` block and a failed `reservation_result`; the modern response returns only the error block plus minimal `auth_result`/`auth_data`.

Apart from redacted account identifiers and the removal of capture artifacts (embedded HTTP-header comment blocks and a stray trailing character), treat them as unmodified records. The individual records:

- Successful authentication — `status` 5 (older field set).

- Unsuccessful authentication — bad credentials (nested `error_body`, `http_code` 404).
- New-computer auth response — pending auth 202.
- New-computer progress response — OTP required (`status` 10).
- New-computer OTP response — OTP submit 202.
- Out-of-date client — 409 / 5008 (with `message` + `ttl_seconds` payload, unlike the modern shape).
- Public realms response — realm listing (2016 field set).

Appendix A.

Request/Response Captures

The complete capture files referenced in §11, reproduced in full. This appendix contains a complete transcript of two logins — a US success and an EU OTP-required flow with an invalid realm — followed by the individual response captures and the older 2016 records. XML content is shown as captured (prettified for readability).

Sanitisation. The embedded transcript has been sanitised — session UUIDs are replaced with `032ef38a-d35d-11eb-8b87-d7f72a84b0bb`, device IDs with a 128-zero placeholder, and account names with `MyUsername`; XML responses are prettified. The individual capture files shown via listings are sanitised the same way: account names and email addresses become `MyUsername` and `eso@example.com`, and user IDs become placeholder digits; session UUIDs there are the captured originals, only the transcript replaces them. Request lines and Host headers in the transcript are normalised from the proxy-capture form (`127.0.0.1:18080`) to the real service host and path, and incidental headers such as `Content-Length` are omitted. These files are non-normative.

The two transcripts. The appendix opens with a clean US login: the client submits credentials, receives 202 Accepted with a session UUID, polls `/login_queue/progress`, and is answered with a `status` 5 success payload carrying `auth_result`, `auth_data`, `reservation_result`, and `eta`. The second transcript repeats the login against the EU service while requesting the NA realm id: the server accepts it, challenges the device with an OTP (`status` 10), and after the OTP is submitted the reservation fails with `status` 6 — the requested realm is not advertised there.

A.1. Successful Auth, US Megaserver

`POST /login_queue/auth HTTP/1.1`

Header	Value
Host	live-services.elderscrollsonline.com
User-Agent	eso/7.0.6.2256692 (live; public; client; win)
Content-Type	application/x-www-form-urlencoded
Accept	application/xml

Form Data	
Key	**Value**
email_address	MyUsername
password	MyPassword
realm_id	4000
device_id	00000000....00000000 (128 zeros)
trusted_machine	1
language	en
hardware_type	1
streaming_status	1

`HTTP/1.1 202 Accepted`

Header	**Value**
Content-Type	application/xml; charset=utf-8
Date	Thu, 17 Jun 2021 09:48:44 GMT
Server	nginx
Set-Cookie	BIGipServerlive-platlogin.live.web. dfw_http_8000_pool=642257930.16415.0000; path=/; Httponly; Secure

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <queue_eta_sec>0.081807782</queue_eta_sec>
    <uuid>032ef38a-d35d-11eb-8b87-d7f72a84b0bb</uuid>
    <callback_interval_ms>2000</callback_interval_ms>
    <queue_position>1</queue_position>
  </response>
  <result_code>2000</result_code>
  <result_message>Authentication Pending</result_message>
</zos_platform_response>
```

POST /login_queue/progress HTTP/1.1

Header	Value
Host	live-services.elderscrollsonline.com
User-Agent	eso/7.0.6.2256692 (live; public; client; win)
Content-Type	application/x-www-form-urlencoded
Accept	application/xml

Form Data	
Key	**Value**
uuid	032ef38a-d35d-11eb-8b87-d7f72a84b0bb
device_id	00000000....00000000 (128 zeros)

HTTP/1.1 200 OK

Header	Value
Content-Type	application/xml; charset=utf-8
Date	Thu, 17 Jun 2021 09:48:47 GMT
Server	nginx
Set-Cookie	BIGipServerlive-platlogin.live.web. dfw_http_8000_pool=2403865610.16415.0000; path=/; Httponly; Secure

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <status>5</status>
    <state_data>
      <data>
        <auth_result>
          <uuid>032ef38a-d35d-11eb-8b87-d7f72a84b0bb</uuid>
          <access_flags>0</access_flags>
          <entitlements_mask>154665472</entitlements_mask>
          <entitlements>10</entitlements>
          <entitlements>22</entitlements>
          <entitlements>21</entitlements>
          <entitlements>28</entitlements>
          <entitlements>25</entitlements>
          <entitlements>20</entitlements>
```

```
            <account_name>MyUsername</account_name>
            <country>US</country>
            <user_id>22222222</user_id>
            <email>eso@example.com</email>
            <sg_new_ip>true</sg_new_ip>
            <console_id>0</console_id>
            <console_subscription>false</console_subscription>
            <is_transfer>false</is_transfer>
            <master_account_id>11111111</master_account_id>
            <game_account_type_ids>23</game_account_type_ids>
            <platform>pc</platform>
          </auth_result>
          <auth_data>
            <email_address>MyUsername</email_address>
            <language>en</language>
            <realm_id>4000</realm_id>
            <ps4_auth_code></ps4_auth_code>
            <hardware_type>1</hardware_type>
            <streaming_status>1</streaming_status>
            <psn_client_region></psn_client_region>
          </auth_data>
          <reservation_result>
            <uuid>032ef38a-d35d-11eb-8b87-d7f72a84b0bb</uuid>
            <succeeded>true</succeeded>
            <realm_name>NA Megaserver</realm_name>
            <connectPort>24507</connectPort>
            <connectAddress>198.20.200.162</connectAddress>
            <realm_id>4000</realm_id>
            <depot_id>4000</depot_id>
            <retry>false</retry>
          </reservation_result>
          <eta>
            <average_wait>0</average_wait>
            <queue_size>0</queue_size>
            <time_last>1.623774182e+09</time_last>
          </eta>
        </data>
      </state_data>
    </response>
    <result_code>2000</result_code>
    <result_message>Reservation Successful</result_message>
</zos_platform_response>
```

A.2. Successful Auth, EU Megaserver, OTP, Invalid Realm

The realm was accidentally left at 4000 when logging into EU (4001). The captured payload carried a stray trailing `&` in the `auth_result` `uuid` (a proxy artifact); it is omitted here so the listing is well-formed XML.

`POST /login_queue/auth HTTP/1.1`

Header	Value
Host	live-eu-services.elderscrollsonline.com
User-Agent	eso/7.0.6.2256692 (live; public; client; win)
Content-Type	application/x-www-form-urlencoded
Accept	application/xml

Form Data	
Key	**Value**
email_address	MyUsername
password	MyPassword
realm_id	4000
device_id	00000000....00000000 (128 zeros)
trusted_machine	1
language	en
hardware_type	1
streaming_status	1

`HTTP/1.1 202 Accepted`

Header	Value
Content-Type	application/xml; charset=utf-8
Date	Thu, 17 Jun 2021 09:50:54 GMT
Server	nginx
Set-Cookie	BIGipServerlive-eu-platlogin.live.web. fra_http_8000_pool=3064075786.16415.0000; path=/; Httponly; Secure

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
```

```
  <response>
    <queue_eta_sec>0.010248611</queue_eta_sec>
    <uuid>032ef38a-d35d-11eb-8b87-d7f72a84b0bb</uuid>
    <callback_interval_ms>2000</callback_interval_ms>
    <queue_position>1</queue_position>
  </response>
  <result_code>2000</result_code>
  <result_message>Authentication Pending</result_message>
</zos_platform_response>
```

POST /login_queue/progress HTTP/1.1

Header	Value
Host	live-eu-services.elderscrollsonline.com
User-Agent	eso/7.0.6.2256692 (live; public; client; win)
Content-Type	application/x-www-form-urlencoded
Accept	application/xml

Form Data	
Key	**Value**
uuid	032ef38a-d35d-11eb-8b87-d7f72a84b0bb
device_id	00000000....00000000 (128 zeros)

HTTP/1.1 200 OK

Header	Value
Content-Type	application/xml; charset=utf-8
Date	Thu, 17 Jun 2021 09:50:56 GMT
Server	nginx
Set-Cookie	BIGipServerlive-eu-platlogin.live.web. fra_http_8000_pool=2929858058.16415.0000; path=/; Httponly; Secure

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
    <response>
        <status>10</status>
        <state_data></state_data>
        <otp_ttl_sec>86400</otp_ttl_sec>
```

```
        <queue_eta_sec>0</queue_eta_sec>
        <callback_interval_ms>1000</callback_interval_ms>
        <queue_position>0</queue_position>
    </response>
    <result_code>2000</result_code>
    <result_message>OTP Waiting</result_message>
</zos_platform_response>
```

`POST /login_queue/otp HTTP/1.1`

Header	Value
Host	live-eu-services.elderscrollsonline.com
User-Agent	eso/7.0.6.2256692 (live; public; client; win)
Content-Type	application/x-www-form-urlencoded
Accept	application/xml

Form Data	
Key	**Value**
otp	ABCDEFGH
device_id	00000000....00000000 (128 zeros)
uuid	032ef38a-d35d-11eb-8b87-d7f72a84b0bb

`HTTP/1.1 202 Accepted`

Header	Value
Content-Type	application/xml; charset=utf-8
Date	Thu, 17 Jun 2021 09:51:09 GMT
Server	nginx
Set-Cookie	BIGipServerlive-eu-platlogin.live.web. fra_http_8000_pool=2929858058.16415.0000; path=/; Httponly; Secure

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <uuid>a50fd878-a253-4331-8677-2e3c90622584</uuid>
    <queue_eta_sec>0.01327475</queue_eta_sec>
    <callback_interval_ms>2000</callback_interval_ms>
    <queue_position>1</queue_position>
```

```
    <status>4</status>
  </response>
  <result_code>2000</result_code>
  <result_message>Reservation Waiting</result_message>
</zos_platform_response>
```

`POST /login_queue/progress HTTP/1.1`

Header	Value
Host	live-eu-services.elderscrollsonline.com
User-Agent	eso/7.0.6.2256692 (live; public; client; win)
Content-Type	application/x-www-form-urlencoded
Accept	application/xml

Form Data	
Key	**Value**
uuid	032ef38a-d35d-11eb-8b87-d7f72a84b0bb
device_id	00000000....00000000 (128 zeros)

`HTTP/1.1 200 OK`

Header	Value
Content-Type	application/xml; charset=utf-8
Date	Thu, 17 Jun 2021 09:51:11 GMT
Server	nginx
Set-Cookie	BIGipServerlive-eu-platlogin.live.web.fra_http_8000_pool=3198293514.16415.0000; path=/; Httponly; Secure

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <status>6</status>
    <state_data>
      <data>
        <auth_result>
          <uuid>032ef38a-d35d-11eb-8b87-d7f72a84b0bb</uuid>
          <access_flags>0</access_flags>
          <entitlements_mask>154665472</entitlements_mask>
```

```
            <entitlements>10</entitlements>
            <entitlements>22</entitlements>
            <entitlements>21</entitlements>
            <entitlements>28</entitlements>
            <entitlements>25</entitlements>
            <entitlements>20</entitlements>
            <account_name>MyUsername</account_name>
            <country>US</country>
            <user_id>22222222</user_id>
            <email>eso@example.com</email>
            <sg_new_ip>true</sg_new_ip>
            <console_id>0</console_id>
            <console_subscription>false</console_subscription>
            <is_transfer>false</is_transfer>
            <master_account_id>11111111</master_account_id>
            <game_account_type_ids>23</game_account_type_ids>
            <platform>pc</platform>
          </auth_result>
          <auth_data>
            <email_address>MyUsername</email_address>
            <language>en</language>
            <realm_id>4000</realm_id>
            <ps4_auth_code></ps4_auth_code>
            <hardware_type>1</hardware_type>
            <streaming_status>1</streaming_status>
            <psn_client_region></psn_client_region>
          </auth_data>
          <reservation_result>
            <succeeded>false</succeeded>
            <realm_name></realm_name>
            <connectPort>0</connectPort>
            <realm_id>0</realm_id>
            <depot_id>0</depot_id>
            <retry>false</retry>
          </reservation_result>
          <eta>
            <average_wait>0</average_wait>
            <queue_size>0</queue_size>
            <time_last>1.623662013e+09</time_last>
          </eta>
        </data>
      </state_data>
    </response>
    <result_code>2000</result_code>
```

```
55    <result_message>Reservation Failed</result_message>
56  </zos_platform_response>
```

A.3. Login Responses

A.3.1. Successful Login

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <status>5</status>
    <state_data>
      <data>
        <auth_result>
          <uuid>2de06c14-4166-4f66-b3ae-b97d35ed9739</uuid>
          <access_flags>0</access_flags>
          <entitlements_mask>154665472</entitlements_mask>
          <entitlements>10</entitlements>
          <entitlements>22</entitlements>
          <entitlements>21</entitlements>
          <entitlements>28</entitlements>
          <entitlements>25</entitlements>
          <entitlements>20</entitlements>
          <account_name>MyUsername</account_name>
          <country>AU</country>
          <user_id>22222222</user_id>
          <email>eso@example.com</email>
          <sg_new_ip>false</sg_new_ip>
          <console_id>0</console_id>
          <console_subscription>false</console_subscription>
          <is_transfer>false</is_transfer>
          <master_account_id>11111111</master_account_id>
          <game_account_type_ids>23</game_account_type_ids>
          <platform>pc</platform>
        </auth_result>
        <auth_data>
          <email_address>MyUsername</email_address>
          <language>en</language>
          <realm_id>4000</realm_id>
          <ps4_auth_code></ps4_auth_code>
          <hardware_type>1</hardware_type>
          <streaming_status>1</streaming_status>
          <psn_client_region></psn_client_region>
        </auth_data>
        <reservation_result>
          <uuid>2de06c14-4166-4f66-b3ae-b97d35ed9739</uuid>
          <succeeded>true</succeeded>
          <realm_name>NA Megaserver</realm_name>
```

```
          <connectPort>24506</connectPort>
          <connectAddress>198.20.200.24</connectAddress>
          <realm_id>4000</realm_id>
          <depot_id>4000</depot_id>
          <retry>false</retry>
        </reservation_result>
        <eta>
          <average_wait>0</average_wait>
          <queue_size>0</queue_size>
          <time_last>1.623673416e+09</time_last>
        </eta>
      </data>
    </state_data>
  </response>
  <result_code>2000</result_code>
  <result_message>Reservation Successful</result_message>
</zos_platform_response>
```

A.3.2. Login Accepted

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <queue_eta_sec>0.009561846</queue_eta_sec>
    <uuid>2de06c14-4166-4f66-b3ae-b97d35ed9739</uuid>
    <callback_interval_ms>2000</callback_interval_ms>
    <queue_position>1</queue_position>
  </response>
  <result_code>2000</result_code>
  <result_message>Authentication Pending</result_message>
</zos_platform_response>
```

A.3.3. Successful Login (8.3.8 era, 2023)

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <status>5</status>
    <state_data>
      <data>
        <auth_result>
          <uuid>c47d337c-2329-47be-a2a1-cad14c009307</uuid>
          <access_flags>0</access_flags>
          <entitlements_mask>0</entitlements_mask>
          <account_name>MyUsername</account_name>
```

```
            <country>AU</country>
            <user_id>22222222</user_id>
            <email>eso@example.com</email>
            <sg_new_ip>true</sg_new_ip>
            <console_id></console_id>
            <console_subscription>false</console_subscription>
            <is_transfer>false</is_transfer>
            <master_account_id>11111111</master_account_id>
            <game_account_type_ids>23</game_account_type_ids>
            <platform>pc</platform>
          </auth_result>
          <auth_data>
            <email_address>MyUsername</email_address>
            <language>en</language>
            <realm_id>4000</realm_id>
            <ps4_auth_code></ps4_auth_code>
            <hardware_type>1</hardware_type>
            <streaming_status>1</streaming_status>
            <psn_client_region></psn_client_region>
          </auth_data>
          <reservation_result>
            <uuid>c47d337c-2329-47be-a2a1-cad14c009307</uuid>
            <succeeded>true</succeeded>
            <realm_name>NA Megaserver</realm_name>
            <connectPort>24503</connectPort>
            <connectAddress>198.20.200.23</connectAddress>
            <realm_id>4000</realm_id>
            <depot_id>4000</depot_id>
            <retry>false</retry>
          </reservation_result>
          <eta>
            <average_wait>0</average_wait>
            <queue_size>0</queue_size>
            <time_last>1.68363982e+09</time_last>
          </eta>
        </data>
      </state_data>
    </response>
    <result_code>2000</result_code>
    <result_message>Reservation Successful</result_message>
</zos_platform_response>
```

A.3.4. Login Accepted (8.3.8 era, 2023)

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<zos_platform_response>
  <response>
    <queue_eta_sec>0.002963025</queue_eta_sec>
    <uuid>c47d337c-2329-47be-a2a1-cad14c009307</uuid>
    <callback_interval_ms>2000</callback_interval_ms>
    <queue_position>1</queue_position>
  </response>
  <result_code>2000</result_code>
  <result_message>Authentication Pending</result_message>
</zos_platform_response>
```

A.3.5. OTP Valid

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <uuid>a50fd878-a253-4331-8677-2e3c90622584</uuid>
    <queue_eta_sec>0.01327475</queue_eta_sec>
    <callback_interval_ms>2000</callback_interval_ms>
    <queue_position>1</queue_position>
    <status>4</status>
  </response>
  <result_code>2000</result_code>
  <result_message>Reservation Waiting</result_message>
</zos_platform_response>
```

A.3.6. Invalid Password

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <status>3</status>
    <state_data>
      <data>
        <error>
          <http_code>401</http_code>
          <result_code>8000</result_code>
          <result_message>Invalid email/passwd</result_message>
          <error_body>{&#34;password&#34;:&#34;incorrect&#34;}</error_b
          ↪  ody>
        </error>
        <auth_result>
          <uuid>be848409-a654-4a22-b914-ed333f6e40c5</uuid>
          <access_flags>0</access_flags>
          <entitlements_mask>0</entitlements_mask>
          <user_id>0</user_id>
          <sg_new_ip>false</sg_new_ip>
          <console_id>0</console_id>
          <console_subscription>false</console_subscription>
          <is_transfer>false</is_transfer>
          <platform>pc</platform>
        </auth_result>
        <auth_data>
          <email_address>MyUsername</email_address>
          <language>en</language>
          <realm_id>4002</realm_id>
```

```
            <ps4_auth_code></ps4_auth_code>
            <hardware_type>1</hardware_type>
            <streaming_status>1</streaming_status>
            <psn_client_region></psn_client_region>
          </auth_data>
        </data>
      </state_data>
    </response>
    <result_code>2000</result_code>
    <result_message>Authentication Failed</result_message>
</zos_platform_response>
```

A.3.7. Bad Cancel

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
    <result_code>11000</result_code>
    <result_message>Invalid queue state Reservation Successful for cancel
    ↪  call: 9f354f6e-ff35-405e-b440-44d3c4a76209</result_message>
</zos_platform_response>
```

A.3.8. Invalid Cancel

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
    <result_code>5003</result_code>
    <result_message>Invalid UUID received:
    ↪  uuid=a9a19d6a-de87-48f8-a48d-2a7ecc3ea564</result_message>
</zos_platform_response>
```

A.3.9. Invalid Reservation

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
    <result_code>11000</result_code>
    <result_message>Invalid queue state InvalidState for progress call:
    ↪  86db3f13-fe08-413a-a8cd-a66f88544463</result_message>
</zos_platform_response>
```

A.4. Realm listings

A.4.1. Public Realm Listing (NA)

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
```

```
  <response>
    <realm>
      <build_changelist>2256692</build_changelist>
      <build_version>6</build_version>
      <ctime>1623923349</ctime>
      <debug_allowed>1</debug_allowed>
      <depot_id>4000</depot_id>
      <entitlements>0</entitlements>
      <entitlements_list></entitlements_list>
      <expires_after_unix_time>1624355349</expires_after_unix_time>
      <live_update_allowed>1</live_update_allowed>
      <lobby_protocol>0</lobby_protocol>
      <lock_state>0</lock_state>
      <major_version>7</major_version>
      <max_population>0</max_population>
      <minor_version>0</minor_version>
      <p4_allowed>1</p4_allowed>
      <population>0</population>
      <protocol_version>6</protocol_version>
      <realm_id>4000</realm_id>
      <realm_name>NA Megaserver</realm_name>
      <realm_rest_server_url></realm_rest_server_url>
      <realm_status>1</realm_status>
      <region_protocol>0</region_protocol>
      <release_allowed>1</release_allowed>
      <require_build_match>0</require_build_match>
      <require_major_match>0</require_major_match>
      <require_minor_match>0</require_minor_match>
      <require_protocol_match>0</require_protocol_match>
      <tool_allowed>1</tool_allowed>
      <world_type>globby</world_type>
    </realm>
  </response>
  <result_code>2000</result_code>
  <result_message>Public Realm Listing</result_message>
</zos_platform_response>
```

A.4.2. Public Realm Listing (NA, JSON)

```
{
  "zos_platform_response": {
    "response": {
      "realm": [
        {
          "build_changelist": 2256692,
```

```
            "build_version": 6,
            "ctime": 1623925959,
            "debug_allowed": 1,
            "depot_id": 4000,
            "entitlements": [0],
            "entitlements_list": "",
            "expires_after_unix_time": 1624357959,
            "live_update_allowed": 1,
            "lobby_protocol": 0,
            "lock_state": 0,
            "major_version": 7,
            "max_population": 0,
            "minor_version": 0,
            "p4_allowed": 1,
            "population": 0,
            "protocol_version": 6,
            "realm_id": 4000,
            "realm_name": "NA Megaserver",
            "realm_rest_server_url": "",
            "realm_status": 1,
            "region_protocol": 0,
            "release_allowed": 1,
            "require_build_match": 0,
            "require_major_match": 0,
            "require_minor_match": 0,
            "require_protocol_match": 0,
            "tool_allowed": 1,
            "world_type": "globby"
          }
        ]
      },
      "result_code": 2000,
      "result_message": "Public Realm Listing"
    }
}
```


A.4.3. Public Realm Listing (EU)


```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <realm>
      <build_changelist>2256692</build_changelist>
      <build_version>6</build_version>
      <ctime>1623923651</ctime>
```

```
        <debug_allowed>1</debug_allowed>
        <depot_id>4000</depot_id>
        <entitlements>0</entitlements>
        <entitlements_list></entitlements_list>
        <expires_after_unix_time>1624355651</expires_after_unix_time>
        <live_update_allowed>1</live_update_allowed>
        <lobby_protocol>0</lobby_protocol>
        <lock_state>0</lock_state>
        <major_version>7</major_version>
        <max_population>0</max_population>
        <minor_version>0</minor_version>
        <p4_allowed>1</p4_allowed>
        <population>0</population>
        <protocol_version>6</protocol_version>
        <realm_id>4001</realm_id>
        <realm_name>EU Megaserver</realm_name>
        <realm_rest_server_url></realm_rest_server_url>
        <realm_status>1</realm_status>
        <region_protocol>0</region_protocol>
        <release_allowed>1</release_allowed>
        <require_build_match>0</require_build_match>
        <require_major_match>0</require_major_match>
        <require_minor_match>0</require_minor_match>
        <require_protocol_match>0</require_protocol_match>
        <tool_allowed>1</tool_allowed>
        <world_type>globby</world_type>
      </realm>
    </response>
    <result_code>2000</result_code>
    <result_message>Public Realm Listing</result_message>
</zos_platform_response>
```

A.4.4. Public Realm Listing (EU, No Session)

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
    <result_code>5001</result_code>
    <result_message>no session Found</result_message>
</zos_platform_response>
```

A.5. Announcements

A.5.1. Multiple Announcements (JSON)

```
{
```

```
  "zos_platform_response": {
    "response": [
      {
        "data": "Welcome to The Elder Scrolls Online and patch 7.0.5 on
        ↪  the North American PC/Mac megaserver.\n \nThe Elder Scrolls
        ↪  Online: Blackwood and Update 30 are now live! Discover the
        ↪  wilds of Blackwood, recruit new Companions, and continue (or
        ↪  begin!) your Gates of Oblivion adventure. To get started,
        ↪  create a new character or travel directly to the Leyawiin
        ↪  Outskirts wayshrine.\nRededicate your Champion Points for
        ↪  free during the Champions Reforged mini-event, starting now
        ↪  and running until June 15.",
        "message_type": "announcement",
        "language": "en"
      },
      {
        "data": "NOTICE: The North American PC/Mac megaserver is
        ↪  currently unavailable while we perform maintenance.\n\n",
        "message_type": "motd",
        "language": "en"
      }
    ],
    "result_code": 2000,
    "result_message": "success"
  }
}
```

A.5.2. Multiple Announcements (XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <data>Welcome to The Elder Scrolls Online and patch 7.0.5 on the
    ↪  North American PC/Mac megaserver.

The Elder Scrolls Online: Blackwood and Update 30 are now live!
↪  Discover the wilds of Blackwood, recruit new Companions, and
↪  continue (or begin!) your Gates of Oblivion adventure. To get
↪  started, create a new character or travel directly to the Leyawiin
↪  Outskirts wayshrine.
Rededicate your Champion Points for free during the Champions Reforged
↪  mini-event, starting now and running until June 15.</data>
    <message_type>announcement</message_type>
    <language>en</language>
```

```
    </response>
    <response>
      <data>NOTICE: The North American PC/Mac megaserver is currently
      ↪  unavailable while we perform maintenance.

</data>
    <message_type>motd</message_type>
    <language>en</language>
  </response>
  <result_code>2000</result_code>
  <result_message>success</result_message>
</zos_platform_response>
```

A.6. Client-error collection

A.6.1. Error Submit

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response></response>
  <result_code>2000</result_code>
  <result_message>success</result_message>
</zos_platform_response>
```

A.6.2. Error Submit (JSON)

```
{
  "zos_platform_response": {
    "response": "",
    "result_code": 2000,
    "result_message": "success"
  }
}
```

A.6.3. Error Submit without `logmessage` parameter

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <result_code>5000</result_code>
  <result_message>Missing Mandatory Parameters:
  ↪  logmessage</result_message>
</zos_platform_response>
```

A.6.4. Error Submit without `logmessage` parameter (JSON)

```
{
  "zos_platform_response": {
    "result_code": 5000,
    "result_message": "Missing Mandatory Parameters: logmessage"
  }
}
```

A.7. Validation errors (live service, 2023)

A.7.1. Invalid Device ID (JSON)

```
{
  "zos_platform_response": {
    "result_code": 5001,
    "result_message": "Invalid Parameter Value: invalid Device ID"
  }
}
```

A.7.2. Malformed Session UUID

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <result_code>5001</result_code>
  <result_message>Invalid Parameter Value: invalid UUID length:
  ↪ 1</result_message>
</zos_platform_response>
```

A.8. Maintenance / out-of-date / bad session

A.8.1. Login Maintenance

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <message>Services are currently unavailable.</message>
    <ttl_seconds>300</ttl_seconds>
  </response>
  <result_code>3000</result_code>
  <result_message>maintenance</result_message>
</zos_platform_response>
```

A.8.2. Login w/ Bad Client Version

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <result_code>5008</result_code>
  <result_message>Client version not supported</result_message>
</zos_platform_response>
```

A.8.3. Login w/ Bad Client Version (JSON)

```
{
  "zos_platform_response": {
    "result_code": 5008,
    "result_message": "Client version not supported"
  }
}
```

A.8.4. Login w/ Bad Session ID

```
<?xml version="1.0" encoding="UTF-8"?>
<zos_platform_response>
  <response>
    <status>1</status>
    <state_data></state_data>
    <otp_ttl_sec>0</otp_ttl_sec>
    <queue_eta_sec>0</queue_eta_sec>
    <callback_interval_ms>10000</callback_interval_ms>
    <queue_position>0</queue_position>
  </response>
  <result_code>2000</result_code>
  <result_message>UUID Not Found</result_message>
</zos_platform_response>
```

A.9. Older-client captures (2016, game patch 2.3.10)

These files are historical 2016 capture records and do **not** describe this server's behaviour (§11).

A.9.1. Successful Auth

```xml
<?xml version="1.0" encoding="utf-8"?>
<zos_platform_response>
  <response>
    <status>5</status>
    <state_data>
      <data>
        <auth_data>
          <realm_id>4000</realm_id>
          <email_address>MyUsername</email_address>
          <ps4_auth_code></ps4_auth_code>
          <language>en</language>
        </auth_data>
        <eta>
          <queue_size>0</queue_size>
          <time_last>1462277947.23</time_last>
          <average_wait>1</average_wait>
        </eta>
        <reservation_result>
          <retry>False</retry>
          <uuid>ca207553-fd05-46c7-9960-9a8c04555142</uuid>
          <succeeded>True</succeeded>
          <realm_name>NA Megaserver</realm_name>
          <connectPort>24507</connectPort>
          <connectAddress>198.20.200.24</connectAddress>
          <realm_id>4000</realm_id>
          <depot_id>False</depot_id>
        </reservation_result>
        <auth_result>
          <master_account_id>11111111</master_account_id>
          <user_id>22222222</user_id>
          <uuid>ca207553-fd05-46c7-9960-9a8c04555142</uuid>
          <entitlements_mask>512</entitlements_mask>
          <legal_docs></legal_docs>
          <country>AU</country>
          <is_transfer>False</is_transfer>
          <console_id>0</console_id>
          <console_entitlements></console_entitlements>
          <email>eso@example.com</email>
```

```
          <access_flags>0</access_flags>
          <sg_new_ip>False</sg_new_ip>
          <console_subscription>False</console_subscription>
          <external_name></external_name>
          <game_account_type_ids>23</game_account_type_ids>
          <account_name>MyUsername</account_name>
        </auth_result>
      </data>
    </state_data>
  </response>
  <result_code>2000</result_code>
  <result_message>Reservation Successful</result_message>
</zos_platform_response>
```

A.9.2. Unsuccessful Auth

```
<?xml version="1.0" encoding="utf-8"?>
<zos_platform_response>
  <response>
    <status>3</status>
    <state_data>
      <data>
        <auth_data>
          <realm_id>4000</realm_id>
          <email_address>MyUsername</email_address>
          <ps4_auth_code></ps4_auth_code>
          <language>en</language>
        </auth_data>
        <eta>
          <queue_size>0</queue_size>
          <time_last>1462289407.77</time_last>
          <average_wait>0</average_wait>
        </eta>
        <reservation_result>
          <retry>False</retry>
          <uuid></uuid>
          <succeeded>False</succeeded>
          <realm_name>0</realm_name>
          <connectPort>0</connectPort>
          <connectAddress></connectAddress>
          <realm_id>0</realm_id>
          <depot_id>False</depot_id>
        </reservation_result>
        <auth_result>
          <master_account_id></master_account_id>
```

```
            <user_id>0</user_id>
            <uuid>43563b0e-0e26-417c-a12f-2b99e3f2f278</uuid>
            <entitlements_mask>0</entitlements_mask>
            <legal_docs></legal_docs>
            <country></country>
            <is_transfer>False</is_transfer>
            <console_id>0</console_id>
            <console_entitlements></console_entitlements>
            <email></email>
            <access_flags>0</access_flags>
            <sg_new_ip>True</sg_new_ip>
            <console_subscription>False</console_subscription>
            <external_name></external_name>
            <account_name></account_name>
          </auth_result>
          <error>
            <error_body>
              <password>incorrect</password>
            </error_body>
            <http_code>404</http_code>
            <result_code>8000</result_code>
            <result_message>Invalid email/passwd</result_message>
          </error>
        </data>
      </state_data>
    </response>
    <result_code>2000</result_code>
    <result_message>Authentication Failed</result_message>
</zos_platform_response>
```

A.9.3. New Device Auth

```
<?xml version="1.0" encoding="utf-8"?>
<zos_platform_response>
    <response>
      <queue_eta_sec>3.33333333333</queue_eta_sec>
      <uuid>0e8a4444-bf4f-4670-8704-1e6d910d1702</uuid>
      <callback_interval_ms>2000</callback_interval_ms>
      <queue_position>1</queue_position>
    </response>
    <result_code>2000</result_code>
    <result_message>OTP Waiting</result_message>
</zos_platform_response>
```

A.9.4. New Device Progress

```
<?xml version="1.0" encoding="utf-8"?>
<zos_platform_response>
  <response>
    <status>10</status>
    <otp_ttl_sec>86400</otp_ttl_sec>
    <callback_interval_ms>1000</callback_interval_ms>
    <state_data></state_data>
    <queue_eta_sec>0</queue_eta_sec>
    <queue_position>0</queue_position>
  </response>
  <result_code>2000</result_code>
  <result_message>OTP Waiting</result_message>
</zos_platform_response>
```

A.9.5. New Device OTP

```
<?xml version="1.0" encoding="utf-8"?>
<zos_platform_response>
  <response>
    <queue_eta_sec>30</queue_eta_sec>
    <uuid>0e8a4444-bf4f-4670-8704-1e6d910d1702</uuid>
    <callback_interval_ms>2000</callback_interval_ms>
    <queue_position>0</queue_position>
  </response>
  <result_code>2000</result_code>
  <result_message>Reservation Waiting</result_message>
</zos_platform_response>
```

A.9.6. Out-of-date Client

```
<?xml version="1.0" encoding="utf-8"?>
<zos_platform_response>
  <response>
    <ttl_seconds>300</ttl_seconds>
    <message>Services are currently unavailable.</message>
  </response>
  <result_code>5008</result_code>
  <result_message>Client version not supported</result_message>
</zos_platform_response>
```

A.9.7. Public Realm Listing

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<zos_platform_response>
  <response>
    <realm>
      <require_major_match>0</require_major_match>
      <world_type>globby</world_type>
      <realm_rest_server_url></realm_rest_server_url>
      <max_population>0</max_population>
      <realm_name>NA Megaserver</realm_name>
      <realm_status>1</realm_status>
      <build_version>10</build_version>
      <p4_allowed>1</p4_allowed>
      <live_update_allowed>1</live_update_allowed>
      <release_allowed>1</release_allowed>
      <minor_version>3</minor_version>
      <debug_allowed>1</debug_allowed>
      <entitlements>0</entitlements>
      <protocol_version>6</protocol_version>
      <lock_state>0</lock_state>
      <require_build_match>0</require_build_match>
      <require_minor_match>0</require_minor_match>
      <expires_after_unix_time>1462977190</expires_after_unix_time>
      <lobby_protocol>0</lobby_protocol>
      <population>0</population>
      <require_protocol_match>0</require_protocol_match>
      <ctime>1462545190</ctime>
      <major_version>2</major_version>
      <region_protocol>0</region_protocol>
      <tool_allowed>1</tool_allowed>
      <realm_id>4000</realm_id>
      <depot_id>4000</depot_id>
    </realm>
  </response>
  <result_code>2000</result_code>
  <result_message>Public Realm Listing</result_message>
</zos_platform_response>
```

Appendix B.

Example Proxy

The interception described in §1.2.1 can be implemented with a small standalone HTTP proxy, and this appendix reproduces one such proxy in full. On start it writes the `Platforms.xml` shown there, extended with the `Proxy (NA)` / `Proxy (EU)` entries that point the client at the local proxy. Each path-prefixed `login_service_url` is then reverse-proxied to the corresponding upstream service.

The proxy has two moving parts. `generatePlatforms` produces the `Platforms.xml` the client reads on startup; the `Proxy (NA)` / `Proxy (EU)` entries point the client at `http://127.0.0.1:8080/us` and `.../eu` instead of the real hosts. `makeHandler` is the forwarding half: each request under `/us`, `/eu` or `/custsupt` is proxied to the matching upstream with the prefix stripped and the `Host` header rewritten. A request whose `User-Agent` is missing or does not start with `eso/` gets a real client `User-Agent` substituted — the live service rejects unparseable agents with HTTP 409/5008 (§5.10), so a plain HTTP client such as `curl` would be refused without it. The substituted agent is the 2023-era client string `eso/8.3.8.2669019 (live; public; client; win)`.

The program uses only the Go standard library. It is an example implementation, not part of the protocol.

```
package main

import (
  "encoding/xml"
  "flag"
  "fmt"
  "net/http"
  "net/http/httputil"
  "net/url"
  "os"
  "strings"
)

const (
  NAHost       = "live-services.elderscrollsonline.com"
  EUHost       = "live-eu-services.elderscrollsonline.com"
  CustHost     = "api-prod.cs.bethesda.net"
  UserAgent    = "eso/8.3.8.2669019 (live; public; client; win)"
  PlatformsXML = "Platforms.xml"
)

func generatePlatforms(filename string, hostPort string) error {
```

```
23    type platform struct {
24      Name               string `xml:"name"`
25      LoginServiceUrl    string `xml:"login_service_url"`
26      CustomerServiceUrl string `xml:"customer_service_url"`
27      Default            bool   `xml:"default"`
28      DefaultRealmId     uint   `xml:"default_realm_id"`
29    }
30
31    type platforms struct {
32      XMLName   interface{} `xml:"platforms"`
33      Platforms []platform  `xml:"platform"`
34    }
35
36    if filename == "" {
37      return nil
38    }
39
40    custUrl := (&url.URL{Scheme: "http", Host: hostPort, Path: "/custsupt/prod"}).String()
41    rawplat, err := xml.MarshalIndent(platforms{Platforms: []platform{
42      {
43        Name:               "Live",
44        LoginServiceUrl:    "https://live-services.elderscrollsonline.com",
45        CustomerServiceUrl: "https://api-prod.cs.bethesda.net/prod",
46        Default:            false,
47        DefaultRealmId:     4000,
48      },
49      {
50        Name:               "Live-EU",
51        LoginServiceUrl:    "https://live-eu-services.elderscrollsonline.com",
52        CustomerServiceUrl: "https://api-prod.cs.bethesda.net/prod",
53        Default:            false,
54        DefaultRealmId:     4001,
55      },
56      {
57        Name:               "Proxy (NA)",
58        LoginServiceUrl:    (&url.URL{Scheme: "http", Host: hostPort, Path:
          ↪  "/us"}).String(),
59        CustomerServiceUrl: custUrl,
60        Default:            false,
61        DefaultRealmId:     4000,
62      },
63      {
64        Name:               "Proxy (EU)",
65        LoginServiceUrl:    (&url.URL{Scheme: "http", Host: hostPort, Path:
          ↪  "/eu"}).String(),
66        CustomerServiceUrl: custUrl,
67        Default:            false,
68        DefaultRealmId:     4001,
69      },
70    }}, "", "    ")
71
72    if err != nil {
73      return err
74    }
75
```

```
76    return os.WriteFile(filename, append([]byte(xml.Header), rawplat...), 0644)
77  }
78
79  func makeHandler(prefix string, host string, userAgent string) http.Handler {
80    return &httputil.ReverseProxy{Rewrite: func(req *httputil.ProxyRequest) {
81      out := req.Out
82
83      out.URL.Scheme = "https"
84      out.URL.Host = host
85      out.URL.Path = strings.TrimPrefix(out.URL.Path, prefix)
86
87      out.Header.Set("Host", host)
88      out.Host = host
89
90      if ua, ok := out.Header["User-Agent"]; !ok || !strings.HasPrefix(ua[0], "eso/") {
91        out.Header.Set("User-Agent", userAgent)
92      }
93    }}
94  }
95
96  func main() {
97    userAgent := flag.String("user-agent", UserAgent, "user agent")
98    platformXML := flag.String("platform-file", PlatformsXML, "Platforms.xml output file
      ↪  (empty to disable)")
99    listenAddress := flag.String("listen-address", "127.0.0.1:8080", "http listen
      ↪  address")
100   flag.Parse()
101
102   if err := generatePlatforms(*platformXML, *listenAddress); err != nil {
103     fmt.Printf("WARNING: unable to generate Platforms.xml: %v", err)
104   }
105
106   mux := http.NewServeMux()
107   mux.Handle("/us/", makeHandler("/us", NAHost, *userAgent))
108   mux.Handle("/eu/", makeHandler("/eu", EUHost, *userAgent))
109   mux.Handle("/custsupt/", makeHandler("/custsupt", CustHost, *userAgent))
110
111   if err := http.ListenAndServe(*listenAddress, mux); err != nil {
112     panic(err)
113   }
114 }
```