



Integrace se Secured Device Gateway API

Dokumentace pro vývojáře

DATUM A VERZE DOKUMENTU

15.7.2025

Verze 1.2

Preamble

Tento dokument vzniknul na základě objednávky ze strany Brněnské komunikace a.s. (BKOM) pod číslem VO-2025-300-000238 v rámci Smlouvy o poskytování služeb na Servisní podpory C-ITS Brno.

Cílem tohoto dokumentu je definice a popis rozhraní SDGW (Secured Device Gateway API) umožňující komunikaci externích systémů a zařízení (např. zábrany, C-ITS jednotky apod.) s centrálním C-ITS Backoffice modulem ACCM, který BKOM provozuje.

Modul ACCM umožňuje vzdálenou správu a management vjezdu do oblastí využívajícího automatického zádržného systému (AZS) v Brněnské aglomeraci.

Nedílnou součástí tohoto dokumentu je také specifikace rozhraní SDGW ve formátu Proto distribuovaná formou ZIP balíčku v jeho aktuálně vydané verzi.

Obsah

PREAMBULE.....	2
1 O DOKUMENTU	5
1.1 HISTORIE VERZÍ	5
1.2 SEZNAM POUŽITÝCH ZKRATEK.....	5
2 TECHNICKÉ VLASTNOSTI ROZHRANÍ SDGW.....	6
2.1 API ROZHRANÍ.....	6
2.1.1 Specifikace rozhraní služeb	6
2.1.2 gRPC	6
2.1.3 Limity požadavků rozhraní SDGW.....	6
2.2 NÁVRATOVÉ STAVY API.....	7
2.3 PATTERNY API.....	7
2.3.1 Outbox pattern	7
2.3.2 Verzování pomocí ETag.....	9
2.3.3 Kontejnerizace pro parciální aktualizace dat.....	10
3 PŘEDPOKLÁDANÝ PROCES INTEGRACE.....	11
3.1 ZAVEDENÍ SMLOUVY DO SYSTÉMU A POVĚŘENÍ DODAVATELE	11
3.2 DODÁVKA A KONFIGURACE APLIKACÍ, SLUŽEB A ZAŘÍZENÍ.....	12
3.3 SCHVÁLENÍ APLIKACÍ, SLUŽEB A ZAŘÍZENÍ PRO KOMUNIKACI PROSTŘEDNICTVÍM ROZHRANÍ SDGW	12
3.4 SPECIFIKACE PŘIPOJENÝCH ZAŘÍZENÍ	12
3.5 VALIDACE DODÁVKY	13
3.6 AKCEPTACE.....	13
4 FUNKČNÍ POŽADAVKY INTEGRACE.....	14
4.1 AUTORIZACE ENROLLMENT IDENTITY	14
4.1.1 Co je Enrollment Identity.....	15
4.1.2 Proces enrollmentu	15
4.1.3 Enrollment a získání certifikátů pro komunikaci s rozhraním SDGW	16
4.2 AUTENTIZACE A SPRÁVA TOKENŮ	29
4.2.1 Autorizované identifikátory	29
4.2.2 Challenge data	31
4.2.3 Získání auth tokenu.....	31
4.2.4 Obnova auth tokenu	32
4.2.5 Využití auth tokenů k autentizaci a autorizaci	33
4.2.6 Obnova certifikátu.....	33
4.2.7 Řešení potíží	34
4.2.8 Seznam dostupných metod služby EA.....	34
4.2.9 Oprávnění.....	35
4.2.10 Registrace nových entit.....	36

4.2.11	Autentizace.....	36
4.2.12	Pravidla pro nakládání s auth tokenem.....	36
5	PROVOZNÍ A TECHNICKÉ POŽADAVKY NA INTEGROVANÉ SYSTÉMY	38
5.1	POŽADAVKY NA INTEGROVANÝ SYSTÉM.....	38
5.2	AUTENTIZACE, AUTORIZACE A DŮVĚRA.....	39
5.3	HSM	39
5.4	AUDITNÍ LOGY	39
5.5	BEZPEČNOSTNÍ INCIDENTY	39
5.6	SERVISNÍ ZÁSAHY.....	40
5.7	ŘEŠENÍ ROZPORŮ MEZI SPECIFIKACÍ ROZHRANÍ SDGW A TÍMTO DOKUMENTEM	40
6	INTEGRACE S MODULEM PRO ŘÍZENÍ PŘÍSTUPU (ACCM)	41
6.1	REGISTRACE INFRASTRUKTURNÍCH ZAŘÍZENÍ.....	41
6.1.1	Registrace přístupových zařízení.....	41
6.1.2	Registrace detekčních zařízení	42
6.2	ZASÍLÁNÍ PROVOZNÍCH STAVŮ	42
6.2.1	Zasílání provozního stavu zařízení.....	43
6.2.2	Zasílání stavu přístupové zábrany.....	43
6.3	ZASÍLÁNÍ DETEKČÍ OBJEKTŮ	44
6.3.1	Zasílání základní detekce objektu.....	44
6.3.2	Doplnění identifikátoru k detekci.....	45
6.3.3	Připojení souboru k detekci.....	45
6.4	OVLÁDÁNÍ ZAŘÍZENÍ POMOCÍ POVELŮ.....	46
6.4.1	Definice podporovaných povelů	46
6.4.2	Přijem povelů ze systému ACCM	46
6.4.3	Zasílání výsledků zpracovaných povelů	47
6.5	ZASÍLÁNÍ INFORMACÍ O OVLÁDÁNÍ.....	47
6.6	ZÍSKÁVÁNÍ KONFIGURACE DETEKČNÍCH ZÓN	48
6.7	ZÍSKÁVÁNÍ KONFIGURACE ČASOVÝCH PLÁNŮ	49
6.8	ZASÍLÁNÍ UDÁLOSTÍ ZE ZAŘÍZENÍ	49
6.9	NAHRÁVÁNÍ SOUBORŮ (NAPŘ. FOTOGRAFIE VOZIDLA)	50

1 O dokumentu

Dokument popisuje základní architekturu a specifikuje technické podmínky pro integraci aplikací, služeb a koncových zařízení se systémem C-ITS Backoffice a jeho moduly prostřednictvím Secured Device Gateway API.

1.1 Historie verzí

Verze	Datum	Popis	Autor
1.0	29. 4. 2024	První verze dokumentu	Jan Novotný
1.1	3. 2. 2025	Upřesnění chování Outbox Patternu	Zuzana Gibová
1.2	15. 7. 2025	Rozšíření o integraci s modulem ACCM	Zuzana Gibová

1.2 Seznam použitých zkratk

Zkratka	Význam
SDGW	Secured Device Gateway API – aplikační programové rozhraní systému pro integraci zařízení a služeb třetích stran
API	Aplikační programové rozhraní
UI	Uživatelské rozhraní
ZD	Zadávací dokumentace
RSA	Technologie pro digitální podpis a šifrování založena na matematické výpočetní obtížnosti faktorizace velkých celých čísel
ECC	Technologie pro digitální podpis a šifrování založená na matematických výpočtech eliptických křivek
EA	Služba <i>Enrollment Authority</i> spravuje požadavky na zapojení aplikací, služeb a zařízení do rozhraní SDGW
EI	<i>Enrollment Identity</i> je entita zaštiťující komunikaci vůči rozhraní SDGW za jedno nebo více zařízení
DevMgr	Služba <i>Device Management Service</i> zajišťuje správu zařízení integrovaných s rozhraním SDGW
CSR	<i>Certificate Signing Request</i> – požadavek na vydání certifikátu
HSM	Hardware Security Module
SSL	Secure Sockets Layer – šifrování síťové komunikace na úrovni transportní vrstvy

2 Technické vlastnosti rozhraní SDGW

V této kapitole jsou popsány základní technické aspekty a požadavky na technologie nutné pro implementaci API a použité patterny při komunikaci.

2.1 API rozhraní

Rozhraní SDGW je přístupné pomocí protokolu **gRPC** (<https://grpc.io/>) s podporou výměny dat ve formátu **Protobuf** (<https://developers.google.com/protocol-buffers>). Protokol se vyznačuje efektivitou a typovým rozhráním. Využití jiného protokolu není z technických a optimalizačních důvodů možné.

2.1.1 Specifikace rozhraní služeb

Integrátor zařízení, která mají být integrována s rozhráním SDGW, získá generovanou API specifikaci ve formátu **Proto** verze proto3 (<https://developers.google.com/protocol-buffers/docs/proto3>).

Ze specifikace **Protofile** je možné vygenerovat klientskou knihovnu v různých programovacích jazycích se všemi potřebnými objekty (**DTO**), a jako takovou je možné ji používat pro komunikaci s vybraným API.

2.1.2 gRPC

gRPC je protokol na bázi HTTP/2, který nabízí následující formy komunikace:

- **Unární** – volání typu **request/response** s typovým **DTO** definovaným pro danou metodu
- **Server streaming** – prvním voláním klienta je otevřen kanál na server (**request**), následně jsou zasílány typové zprávy (dle **DTO** definovaného pro danou metodu) ze serveru na klienta, a to do doby, než je kanál z jedné nebo druhé strany uzavřen. Klient v tomto režimu zprávy (vyjma úvodního **requestu**) nezasílá.
- **Client streaming** – prvním voláním klienta je otevřen kanál na server (**request**), následně jsou zasílány typové zprávy (dle **DTO** definovaného pro danou metodu) z klienta na server, a to do doby, než je kanál z jedné nebo druhé strany uzavřen. Server v tomto režimu zprávy nezasílá.
- **Duplex streaming** – prvním voláním klienta je otevřen kanál na server (**request**). Typové zprávy (dle **DTO** definovaného pro danou metodu) jsou následně zasílány oběma směry, a to do doby, než je kanál z jedné nebo druhé strany uzavřen. Pro spárování **requestu** a **responsu** je v tomto typu komunikace využíván atribut **request_id**.

2.1.3 Limity požadavků rozhraní SDGW

Komunikace s rozhráním SDGW je monitorována, včetně množství provedených požadavků jednotlivých typů. V některých případech je aplikován **rate limit**.

Integrátoři musí při implementaci zohlednit návratové stavy, kterými API indikuje chybový stav, nebo to, že jej daný klient kontaktuje příliš často (porušení **rate limitu**).

Příkladem omezení mohou být unární volání, která nejsou určena pro častou aktualizaci dat a nesmějí být k tomuto účelu zneužívána. Dále např. opakovaná odesílání totožných dat, na která již rozhraní SDGW jednou odpovědělo chybovým stavem.

Takové situace jsou považovány za defekt a dodavatel nebo provozovatel takového systému je povinen defektní chování odstranit. Do doby, než bude chování upraveno, je toto zařízení z pohledu systému považováno za nefunkční.

V případě trvajících a dlouhodobě neřešeného škodlivého chování může být konkrétní klient nebo systém omezen, či zcela odpojen.

2.2 Návrátové stavy API

SDGW vrací návratové stavy v hlavičce odpovědi. Výčet návratových stavů a jejich užití je možné nalézt na adrese https://grpc.github.io/grpc/core/md_doc_statuscodes.html.

V případě chyby validace vstupního atributu je vrácena chybová hodnota `INVALID_ARGUMENT`, která je však rozšířena o výčet atributů, u kterých se validace nezdařila. Návratový stav se pak objeví např. v tomto znění: *Property „název chybně vyplněného atributu“ failed validation*.

2.3 Patterny API

Rozhraní SDGW používá při integraci následující obecné patterny:

2.3.1 Outbox pattern

Outbox pattern se používá pro zajištění konzistence dat mezi klientem a serverem. Záznamy o změnách jsou ukládány do Outboxu – chronologicky seřazené fronty specifické pro konkrétního klienta a daný typ dat. Každý záznam v Outboxu má přiřazený index, který označuje jeho pozici ve frontě.

Pokud klient žádá data ze systému poskytovaná rozhraním SDGW prostřednictvím Outbox patternu, musí v požadavku uvést index odpovídající poslední úspěšně zpracované události. Server následně vrátí všechna data od tohoto indexu do současnosti v chronologickém pořadí. Některé metody však nemusí vracet všechny historické záznamy, a to z důvodu optimalizace.

V technické specifikaci API (`proto`) lze Outbox pattern rozpoznat podle použití gRPC metody `duplex stream` a přítomnosti atributu `EventInfo` v odpovědích serveru.

Klient je odpovědný za správné zpracování a perzistentní ukládání přijatých dat. Nesmí index zneužívat, například posíláním konstantní hodnoty nebo jeho záměrným nezasíláním. Takové jednání je logováno a může vést k omezení přístupu k rozhraní SDGW.

2.3.1.1 Příklad implementace klienta

Tento příklad ukazuje, jak klient přijímá změny konfigurace sčítacích zón zařízení pomocí Outbox patternu. Každé zařízení (identifikované `requestedObjectId`) má přiřazený seznam konfigurací, jejichž změny sleduje.

Pseudokód ilustruje proces přijímání zpráv ze serveru a správné chování konzumenta Outbox patternu. Detaily zpracování přijatých dat nejsou součástí kódu a jsou reprezentovány funkcemi `HandleNewOrUpdated` a `HandleDeletion`, které musí integrátor implementovat.

2.3.1.1.1 Popis procesu

1. **Navázání spojení:** Po inicializaci gRPC komunikace klient odesílá první požadavek s `lastConfirmedVersion = 0`, což znamená, že dosud žádná zpráva nebyla zpracována.
2. **Přijímání dat:** Klient přijímá změny od serveru a analyzuje jejich typ (`vytvoření`, `aktualizace`, `smazání`).
3. **Zpracování změn:** Pokud změna obsahuje ID, klient může načíst další detaily. Smazání se zpracovává přímo.
4. **Potvrzení zpracování:** Po úspěšném uložení změny klient potvrzuje přijetí serveru a aktualizuje `lastConfirmedVersion`, který se perzistentně ukládá.

Tento mechanismus zajišťuje, že při opětovném připojení klient pokračuje od poslední potvrzené změny.

Doporučuje se nejprve provést perzistentní uložení změny a teprve poté odeslat potvrzení serveru. Pokud by potvrzení bylo odesláno dříve a následné uložení selhalo, server už ji nemusí být schopen znovu poskytnout, což by vedlo k nekonzistenci dat.

2.3.1.1.2 Pseudokód implementace

```
// Pseudocode for receiving zone configuration changes

private void ListenToZoneConfigurationChanges(authToken)
{
    lastConfirmedVersion = persistentStorage.GetLastZoneConfigurationEventVersion()
    while (true)
    {
        try
        {
            // Prepare gRPC communication
            var grpcCallOptions = {
                headers: {
                    {"Authorization", "Bearer " + authToken}
                }
            };
            var clientCall = grpcClient.SubscribeToZoneConfigurationChanges(grpcCallOptions);

            // Send the first request with the last confirmed event index
            clientCall.RequestStream.Write(new StreamRequest {
                RequestedIds = { requestedObjectId },
                EventVersion = lastConfirmedVersion
            });

            // Listen for incoming changes
            foreach (var receivedChange in clientCall.ResponseStream.ReadAll())
            {
                var observedEvent = receivedChange.EventInfo;

                // Process the received change
                if (receivedChange.EventInfo.EventType is Created or Updated)
                {
                    // Fetch additional details for the incoming ID
                    var receivedDetails = grpcClient.GetDetails(new DetailRequest {
                        Id = receivedChange.Id
                    }, grpcCallOptions);

                    HandleNewOrUpdated(receivedDetails); // Implement handling for new or upda
```



```

ted configurations
    }
    else
    {
        HandleDeletion(receivedChange); // Implement handling for deletions
    }

    // Confirm event processing to the server
    clientCall.RequestStream.Write(new StreamRequest {
        ConfirmedRequestedIds: [{
            Id = requestedObjectId,
            ConfirmedEventVersion = observedEvent.EventVersion }
        ]
    });

    // Persistently store the last confirmed event version
    persistentStorage.SetLastZoneConfigurationEventVersion(lastConfirmedVersion);
}
}
catch (Exception ex)
{
    // Handle errors and potentially reconnect and retry
}
}
}

```

2.3.2 Verzování pomocí ETag

Za účelem zabránění zasílání konkurenčních povelů do zařízení z více aplikací nebo uživatelských účtů současně, rozhraní SDGW podporuje verzování obsahu pomocí ETagů (**etag**).

Funkce je implementována tak, že v případě, kdy zařízení předává data do systému, a datový objekt, který je zasílán, obsahuje atribut **etag**, je očekáváno, že zařízení do něj vyplní unikátní hodnotu, která reprezentuje aktuální verzi obsahu. Pokud se obsah změní, je změněn také **etag**. Rozhraní SDGW poté s každým požadavkem na změnu zasílá také aktuálně evidovanou hodnotu **etag**, která definuje, z jaké datové sady před zasláním aktuální změny vycházel. Zařízení následně porovná, zda je zasláná hodnota atributu shodná s aktuálně evidovanou hodnotou odpovídající danému typu a instanci dat. Pokud ano, změny provede, a rozhraní SDGW zpět potvrdí úspěch. Pokud ne, rozhraní SDGW vrátí chybový stav s informací, že verze **etag** nesouhlasí.

Tímto přístupem je zajištěno, že pouze právě jeden požadavek na změnu dat v jednom okamžiku je skutečně proveden, ostatní jsou odmítnuty, a nedochází tak k neočekávaným přepisům dat z více systémů, aplikací nebo uživatelských účtů přistupujících k totožným informacím.

Stejný pattern může být implementován také na straně rozhraní SDGW. Pokud je tedy ve vrácených datech atribut **etag**, je očekáváno, že s požadavkem na změnu bude hodnota tohoto parametru předána do požadavku (**request**) nebo zasílaného datového objektu (streaming message) ze strany zařízení.

2.3.2.1 Datový typ a mechanismus generování

Atribut **etag** je datového typu **string**. Jeho hodnota může být generována náhodně, hashováním, nebo např. inkrementem čísla datového typu **int64** a následně převedením na hodnotu datového typu **string**.

Pro hodnotu parametru **etag** je nutné nastavit unikátní hodnotu po každé aktualizaci dat v zařízení, a to i v případě, že je změna provedena jiným systémem – např. přímým zásahem operátora (viz výše).

Hodnota atributu **etag** musí být nastavena nebo generována jako unikátní pro každou datovou instanci samostatně. V takovém případě by se totiž **etag** měnil příliš často (jeden **etag** společný pro více typů nebo

větší skupinu dat), kdy by nebylo prakticky možné provést aktualizaci dat, protože by zaslaná data většinou obsahovala již zastaralou hodnotu.

2.3.3 Kontejnerizace pro parciální aktualizace dat

Při výměně dat mezi systémy, neohledě na to, kdo je hlavním zdrojem dat, je vhodné nepřenášet veškerá data v každém požadavku nebo zprávě. V takovém případě je optimální zasílat jen části, které byly změněny nebo které je potřeba v dané chvíli aktualizovat.

Protokol rozhraní SDGW podporuje parciální aktualizaci dat pomocí kontejnerizace. To v praxi znamená, že datový objekt přenášené zprávy je rozdělen na povinné části (např. identifikátory), které je nutné zasílat vždy a **containers**, které mohou, ale nemusí být vyplněny.

Při aktualizacích je nutné dbát následujících pravidel

- Pokud je nějaký kontejner použit, musí mít vyplněny všechny položky pro zabránění neočekávaným změnám a s tím souvisejícím stavům
 - Pokud některá property kontejneru není vyplněna, daná property je na serveru vyplněna výchozí hodnotou a jako takovou je přepsána – pokud tedy nějaká property v datech obsahuje hodnotu, ale při zasílání kontejneru není vyplněna, je při aktualizaci přepsána výchozí hodnotou
 - Pokud je hodnota celého kontejneru (jakožto objektu) **null**, není kontejner vůbec zpracován a hodnoty v datovém úložišti nejsou přepsány
- Vždy je zapotřebí v datech zasílat alespoň jeden vyplněný kontejner – v opačném případě je daný stav považován za defekt integrátora, neohledě na výsledek akceptačních testů – to, že server při vývoji a testování nevrátí chybový stav není považováno za akceptované chování. Budoucí aktualizace systému mohou implementovat striktnější validaci a toto omezení tedy uplatnit kdykoliv později.

3 Předpokládaný proces integrace

Pro uvedení celého kontextu integrace jednotlivých telematických aplikací, služeb a zařízení do systému různými subjekty je v tomto dokumentu předpokládán proces, ve kterém integrace podléhá schvalování každé integrace ze strany Zadavatele, a to na základě uzavřené smlouvy s Dodavatelem (provozovatelem zařízení nebo služeb).

1	2	3
Zadavatel	Dodavatel	Zadavatel
Přiřazení smlouvy v systému organizaci Dodavatele	Vytvoření Enrollment Identity a generování CSR	Schválení Enrollment Identity Zadavatelem
Zadavatel vytvoří v systému záznam o existující smlouvě a tu přiřadí organizaci dodavatele. Od daného okamžiku má Dodavatel přístup k informacím o konfiguraci dle požadované funkcionality.	Dodavatel dodá, nainstaluje a nakonfiguruje dodávané aplikace, služby a zařízení. Vytvoří jednu nebo více Enrollment Identity a pro tyto budou vygenerovány key-pairs, které EI následně použije pro podepsání vygenerovaného CSR. CSR je zasláno do služby Enrollment Authority ke schválení.	Zadavatel obdrží informaci o potřebě schválení nových Enrollment Identities, které validuje v kontextu smlouvy a následně schválí. Schválením jsou vygenerovány certifikáty a tyto jsou zaslány v odpovědi na CSR do zařízení.
4	5	6
Dodavatel	Dodavatel	Zadavatel
Integrace zařízení	Validace Dodavatelem a předání k akceptaci	Validace dodávky a akceptace Zadavatelem
Po schválení Enrollment Identity je možné plně komunikovat se zbytkem systému. Dodavatel zajistí, aby do systému byly zaneseny informace o všech připojených zařízeních a přenášena všechna data vyplývající ze ZD.	Dodavatel zkontroluje, že splnil všechny podmínky dané ZD a smlouvu jako celek v systému označí jako připravenou k validaci Zadavatelem.	Zadavatel zkontroluje, zda jsou naplněny všechny podmínky vyplývající ze Smlouvy a Smlouvu jako celek buďto vrátí do předchozího kroku nebo tuto schválí, a tím dodávku ukončí.

3.1 Zavedení smlouvy do systému a pověření Dodavatele

Proces začíná na straně Zadavatele, konkrétně zavedením smlouvy a přiřazením smlouvy organizaci Dodavatele, který si dále deleguje přístup ke smlouvě na pracovníky (řešitele / uživatelské účty) v rámci své organizace.

3.2 Dodávka a konfigurace aplikací, služeb a zařízení

Díky přiřazení smlouvy na organizaci Zadavatele mají pověření uživatelé, po přihlášení do uživatelského rozhraní systému, přístup k informacím nutným pro konfiguraci a integraci zařízení do systému, zejména pak

- Informacím pro konfiguraci připojení (URL, IP adresy a porty) jednotlivých integrovaných služeb a jejich veřejné certifikáty pro důvěryhodné připojení
- Identifikátory, která kategorizují nebo opravňují aplikace, zařízení a služby ke komunikaci
- Požadavky na enrollment (**Enrollment Requests**) uložených na straně služby **Enrollment Authority (EA)**

Řešitel na straně Dodavatele nakonfiguruje aplikace, služby a zařízení dle informací zobrazených u vybrané smlouvy v uživatelském rozhraní.

Dodávané části řešení (aplikace, služby a zařízení), při prvním spuštění, automatizovaně vytvoří požadavek na vydání certifikátu (CSR). Tento podepíše vlastním klíčem, zašifruje veřejným klíčem **Enrollment Authority** (dále jen „EA“), a následně zašlou na službu EA ke schválení a vydání certifikátu (**Enrollment Request**).

3.3 Schválení aplikací, služeb a zařízení pro komunikaci prostřednictvím rozhraní SDGW

Pracovníci Dodavatele v uživatelském rozhraní systému sledují vznikající požadavky na enrollment (**Enrollment Requests**), které po jejich kontrole, a přiřazení k odpovídající smlouvě, postupně nebo najednou předávají ke schválení Zadavateli.

Zadavatel schválením v administrátorské části systému potvrdí, že zařízení uvedená pod danou **Enrollment Request** jsou součástí dodávky a pro integraci se systémem, prostřednictvím rozhraní SDGW, schválí

- **Enrollment Identity**
- Žádné, jedno, více nebo všechna zařízení spojená s vybranou **Enrollment Identity**

Takto schválené **Enrollment Identity** je vydán certifikát, který je si povinna bezpečně uložit. Schválená Enrollment Identity může komunikovat pouze za zařízení, která jsou k této identitě připojena a také schválena.

Zařízení připojená ke schválené **Enrollment Identity** jsou v tomto okamžiku obvykle schválena v provozním režimu **Test**. Provozní režim nemá vliv na pokračování v integraci. Více informací o provozních režimech zařízení je uvedeno dále v tomto dokumentu.

3.4 Specifikace připojených zařízení

Od okamžiku schválení **Enrollment Identity (EI)** je tato služba oprávněna a schopna registrovat nová zařízení, aktualizovat specifikaci a konfiguraci zařízení připojených k této EI, stejně jako za taková zařízení publikovat (zasílat) data do systému. K tomu je nutné získat autentizační token. Autentizační token služba EI získá voláním služby EA.

3.5 Validace dodávky

Před přesunutím zařízení do pilotního nebo produkčního provozu je nutné, aby Dodavatel ověřil úspěšnost integrace se systémem prostřednictvím rozhraní SDGW. Pro ověření úspěšné integrace a naplnění dodávky Dodavatelem je nutné, aby všechny splňovaly následující

- V rámci smlouvy byla dodána všechna zařízení a související komponenty a služby
- Všechny **EI** a všechna zařízení jsou schválená Zadavatelem
- Všechny **EI** jsou enrollovány do služby **EA**
- Všechna zařízení jsou registrována a průběžně aktualizována, včetně konfigurace zařízení, komponent a provozních stavů. Tato data jsou odesílána do služby **DevMgr**.
- Dle typu dodávaných zařízení
 - Např. data ze sčítačů dopravy jsou úspěšně publikována do centrálního systému
- Všechny části jsou stabilní a v rámci jejich provozu nejsou pozorovány ani měřeny žádné anomálie

Pokud je vše výše naplněno, a předmětná část ZD je tímto splněna, Dodavatel označí smlouvu, a tím všechny její komponenty, jako „Připraveny k akceptaci“.

Pokud již nyní není nastaven vyšší režim provozu, jsou v tomto kroku zařízení automaticky přepnuta do provozního režimu **Stage**.

3.6 Akceptace

Zadavatel validuje všechna kritéria předané dodávky dle ZD. V případě, že je vše v pořádku, smlouvu a všechny její komponenty akceptuje.

Ve výchozím stavu budou zařízení připojená k takové smlouvě přesunuty do provozního režimu **Pilot**. Zadavatel může zvolit jakýkoliv jiný provozní režim, např. **Production** apod.

Část integrace s centrálním systémem je tímto krokem považována za ukončenou.

UPOZORNĚNÍ:

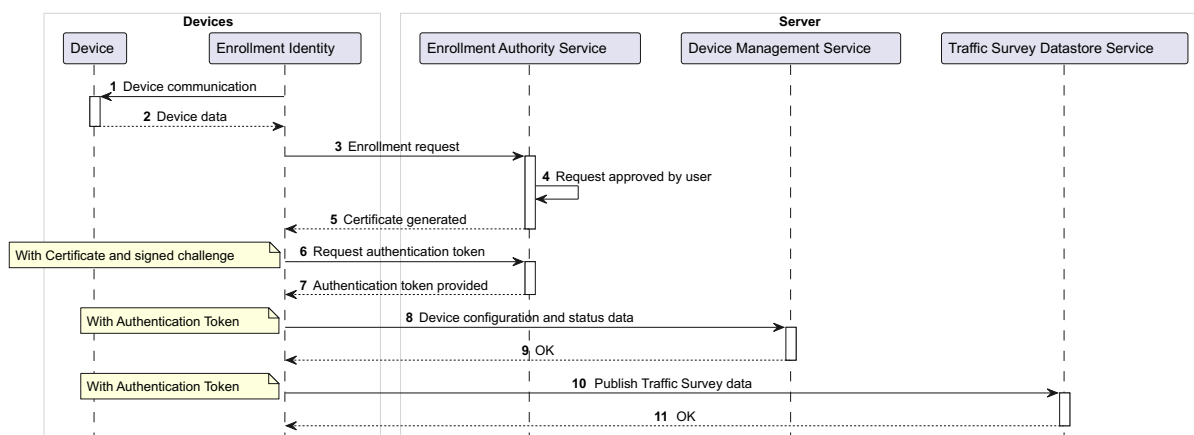
Krok Akceptace učiněný odpovědnou osobou Zadavatele v centrálním systému je pouze administrativním úkonem a **nelze považovat za formální a právně závaznou akceptaci a naplnění smlouvy**. Příznak nastavené Akceptace u smlouvy (dodávky) v systému je nutné chápat pouze jako jednu z podmínek, která musí být splněna pro formální akceptaci smlouvy jako celku. Výčet podmínek je v tomto případě plně v dikci Zadavatele. Tato akce zároveň Dodavatele nevyvazuje z odpovědnosti za vady a další související smluvené nebo zákonem stanovené náležitosti.

4 Funkční požadavky integrace

Pro úspěšnou integraci stávajících nebo nově dodávaných aplikací, služeb a zařízení s centrálním systémem je z pohledu softwarových aplikací, služeb a firmware zařízení potřeba funkčně implementovat následující algoritmické části

1	2	3	4
Autorizace Enrollment Identity	Správa autentizačních tokenů	Správa konfigurace a stavů	Zpracování a publikace dat
Autorizace zařízení, služby nebo aplikace, která umožní zabezpečenou komunikaci se systémem za účelem získávání autentizačních tokenů	Proces pro získávání autentizačních tokenů a automatizované obnovy v okamžiku jejich expirace	Proces pro pravidelnou publikaci konfiguračních, provozních a stavových informací zařízení spravovaných danou Enrollment Identity	Zpracování dat a pravidelná publikace dat do centrálního systému dle povahy jednotlivých zařízení ¹

Zjednodušený náhled na proces integrace s jednotlivými službami a posloupnost jednotlivých kroků je znázorněna na následujícím diagramu.



Zjednodušený proces integrace

4.1 Autorizace Enrollment Identity

Rozhraní SDGW je chráněno proti neoprávněné integraci a komunikaci. Každý systém nebo zařízení, které s tímto rozhraním potřebuje komunikovat přímo, musí být centrálním systémem autorizováno. Tento proces je dále v dokumentu nazýván **enrollment**.

¹ Pro účely této specifikace jsou uvažována data z ASD, procesně je však postup totožný pro publikaci jakýchkoliv jiných typů telematických, stavových a jiných dat. Záleží vždy na obsahu ZD, která typ a rozsah požadovaných dat definuje a na cílové službě, do které budou taková data zasílána.

Zjednodušeným pohledem, proces enrollmentu na straně zařízení vyžaduje vygenerování privátního a veřejného klíče (**key-pair**), vytvoření žádosti o certifikát (**CSR**) a následné uchování certifikátu vráceného rozhraním SDGW na straně zařízení, služby či serveru.

Kritickou částí procesu enrollmentu, a pozdějšího prokázání autenticity enrollovaného zařízení, je bezpečné uchování vygenerovaných klíčů (**key-pair**), získaných certifikátů a případných **auth** tokenů, potřebných pro komunikaci s rozhraním SDGW. Tyto údaje musí být řádně zabezpečeny proti úniku, kopírování, podvržení a další. Ideálně formou **tamper** switche. Únik některé z těchto informací je považováno za závažný bezpečnostní incident a autorizovaná entita může být dočasně nebo trvale vyloučena z možnosti komunikovat s centrálním systémem.

Tato část dokumentu uvádí detailní informace o tom, jak zařízení nebo službu úspěšně enrollovat.

4.1.1 Co je Enrollment Identity

Pokud je vůči rozhraní SDGW enrollováno konkrétní zařízení přímo, zařízení samo za sebe zasílá také stavová a provozní data, případně data dle povahy své funkce (např. sčítací data z ASD), pak pod pojmem **Enrollment Identity (EI)** můžeme chápat toto konkrétní zařízení jako celek, které reprezentuje pouze samo sebe.

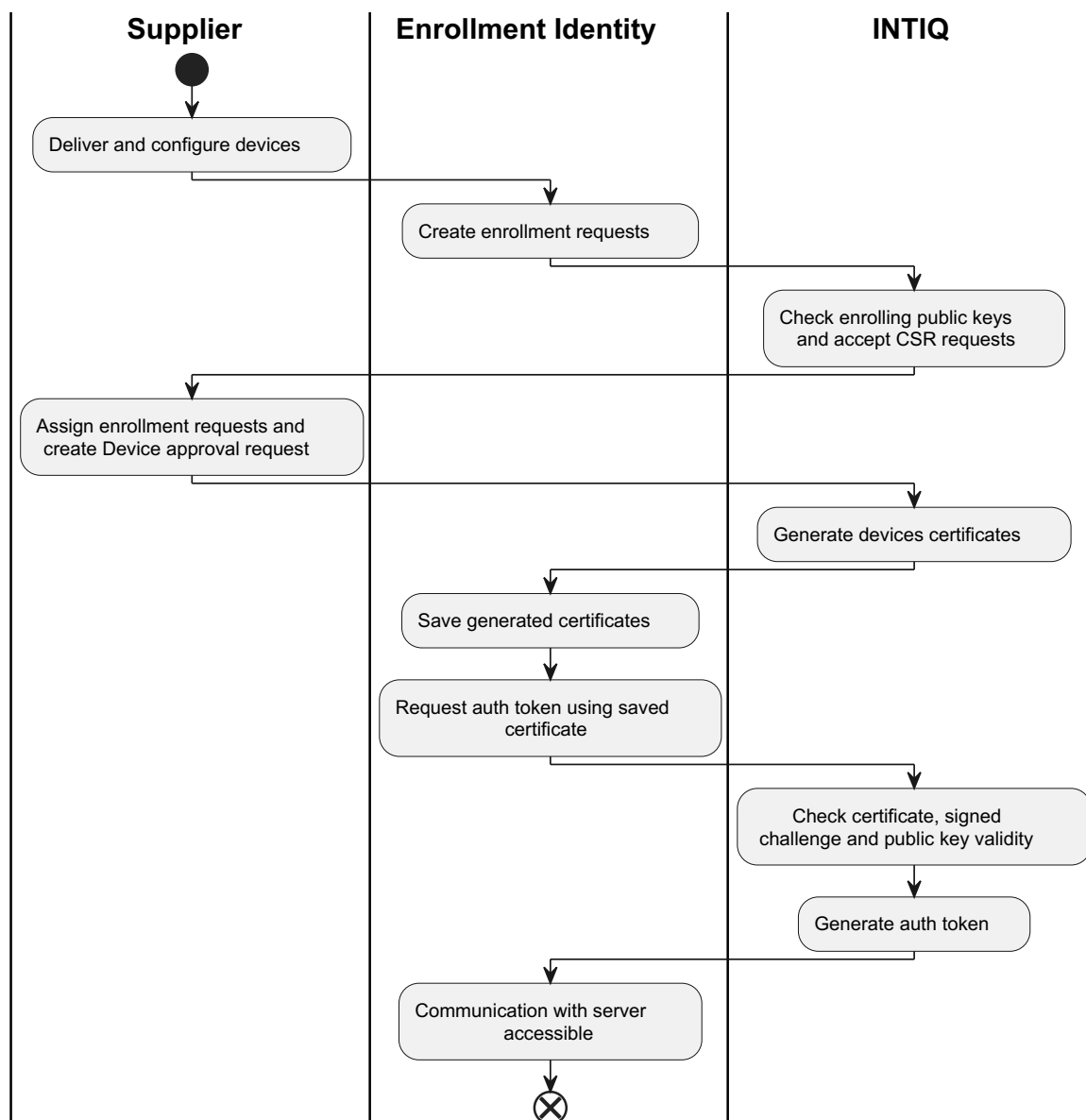
Dalším možným způsobem integrace je enrollment zařízení, služby či aplikace (např. řídicí jednotky nebo řídicího systému), které jsou registrovány v roli **Enrollment Identity (EI)**, ke které je připojeno jedno nebo více dalších zařízení, za které následně **EI** (z pohledu rozhraní SDGW) vystupuje. Taková **EI** pak zasílá specifikace, konfigurace, stavová a provozní data, a dále data dle povahy vícero zařízení (ASD, kamery apod.)

Pojmem **Enrollment Identity (EI)** se tedy rozumí jakákoliv služba, zařízení, případně systém, který je integrován s rozhraním SDGW napřímo, bez prostředníka. **EI** současně reprezentuje jednu nebo více entit a odpovídá tak za správnost, úplnost a konzistenci dat zasílaných do centrálního systému, stejně jako za to, že komunikace odpovídá specifikaci a technickým požadavkům integrace.

Z důvodu zajištění bezpečnosti je důrazně doporučeno, aby **Enrollment Identity** komponenta byla implementována jako přímá součást zařízení. Eliminuje se tak jeden z možných vektorů kybernetických útoků.

4.1.2 Proces enrollmentu

Proces a jeho jednotlivé fáze shrnuje následující diagram.



Sekvenční diagram fází procesu enrollmentu

4.1.3 Enrollment a získání certifikátů pro komunikaci s rozhraním SDGW

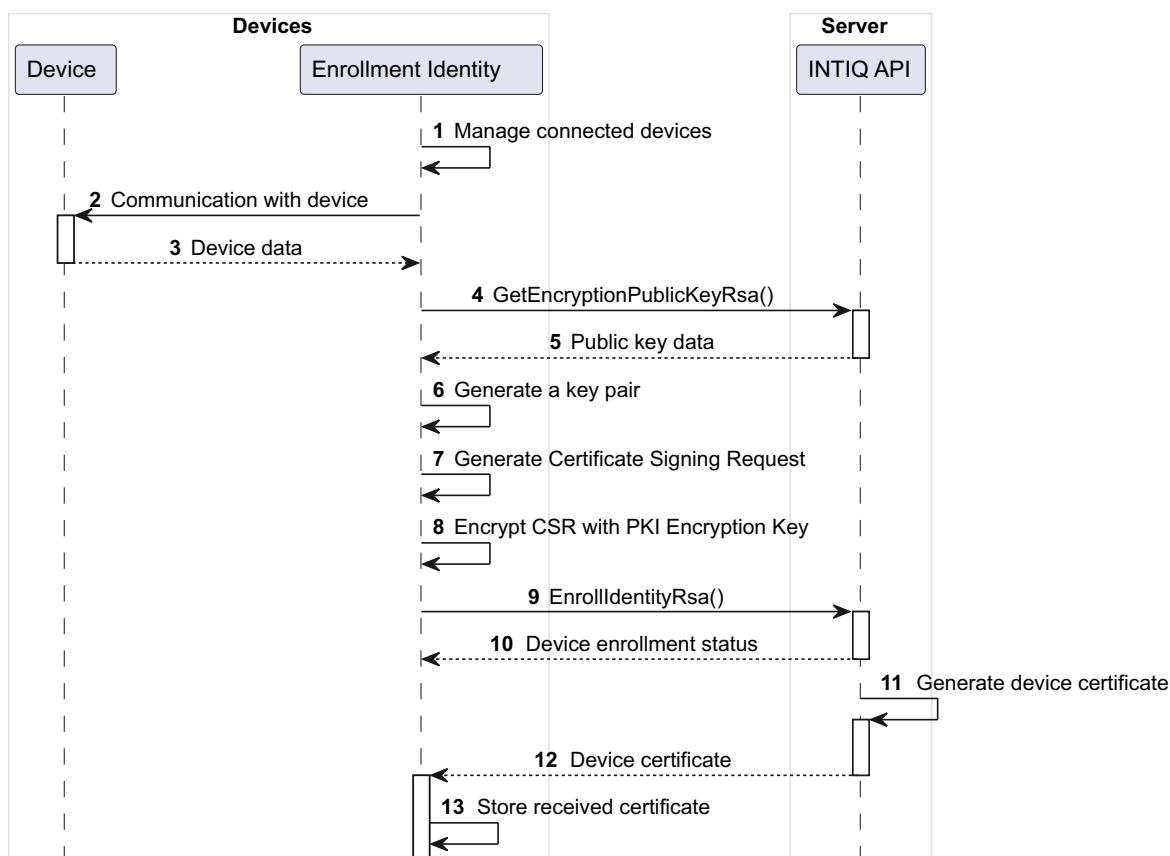
Před vygenerováním žádosti o certifikát (CSR) je nutné získat veřejný klíč rozhraní SDGW. Tento klíč slouží pro ověření i šifrování dat, která se vyměňují mezi EI a tímto rozhraním. Pro získání veřejného klíče je nutné použít jednu z API metod `GetEncryptionPublicKeyRsa` nebo `GetEncryptionPublicKeyEc` (záleží na zvolené metodě šifrování).

Následně je nutné na straně EI vygenerovat dvojici `key-pairs` (`Verification` a `Encryption`) a CSR (`Certificate Signing Request`).

Jakmile jsou kroky výše hotovy, je možné tato data použít pro žádost o enrollment pomocí volání API metody `EnrollIdentityRsa` nebo `EnrollIdentityEc`. Zvolená metoda musí odpovídat metodě šifrování použité při získání veřejného klíče.

Rozhraní SDGW, v případě schválení zařízení, vygeneruje a vrátí tomuto zařízení vygenerovaný digitální certifikát. Tento certifikát je použit v následném kroku pro autentizaci a získání `auth` tokenu, který je nutný pro plnou komunikaci s rozhraním SDGW.

Kroky nutné k autorizaci `Enrollment Identity` jsou vizualizovány na následujícím diagramu s využitím metody šifrování RSA.



Sekvenční diagram autorizace EI

4.1.3.1 Získání veřejného klíče služby Enrollment Authority

I přesto, že komunikace s rozhraním SDGW je šifrována na úrovni transportní vrstvy (`SSL`), vyžadují některé operace dodatečné zabezpečení na úrovni podpisu a šifrování obsahu požadavku. V případech, které pokrývá tento dokument jde o následující operace

- Zaslání `CSR` prostřednictvím rozhraní SDGW pro žádost o enrollment `EI`
- Získání `Challenge` z rozhraní SDGW pro ověření vlastnictví privátního klíče
- Zaslání žádosti o vydání `auth` tokenu rozhraní SDGW

Pro tyto účely si musí **EI** obstarat veřejný **Encryption Key** rozhraní SDGW, pro kterou data šifruje. Tento klíč získá pomocí volání metody **GetEncryptionPublicKeyRsa** či **GetEncryptionPublicKeyEc**. Veřejný klíč je vrácen ve formátu **PKCS #1** u RSA typu šifrování. U šifrování typu ECC (eliptické křivky) je vrácen veřejný klíč pomocí souřadnic bodů X a Y eliptické křivky.

4.1.3.2 Generování key-pairs

Pro možnost výměny klíčů a certifikátů je nutné, aby **EI** měla k dispozici dvě sady klíčů (**key-pair**) – **Verification** a **Encryption**. Pokud tyto klíče **EI** nemá k dispozici, musí je vygenerovat. Klíče jsou potřeba k následujícím akcím

- Digitální podpis vygenerovaného **CSR** (**Verification** key)
- Dešifrování obsahu zasílaného serverem (**Encryption** key)

Pokud **Enrollment Identity** tyto klíče nemá vygenerovány, je povinná je vygenerovat.

Oba páry **key-pairs** musí splňovat následující podmínky

- Odpovídá podporovaným šifrám podporovaných rozhraním SDGW, viz níže
- Je vygenerován zabezpečeně, tj. pokud neexistuje výjimka udělená Zadavatelem, musí být klíče vygenerovány a uloženy uvnitř připojeného **HSM** (Hardware Security Module)
- **EI** má ke **key-pairu** přístup i po restartech a reinicializacích zařízení
- V případě napadení zařízení je klíč nenávratně znehodnocen nebo ztracen (ochrana **Tamper**)

Rozhraní SDGW podporuje klíče ve dvou formátech:

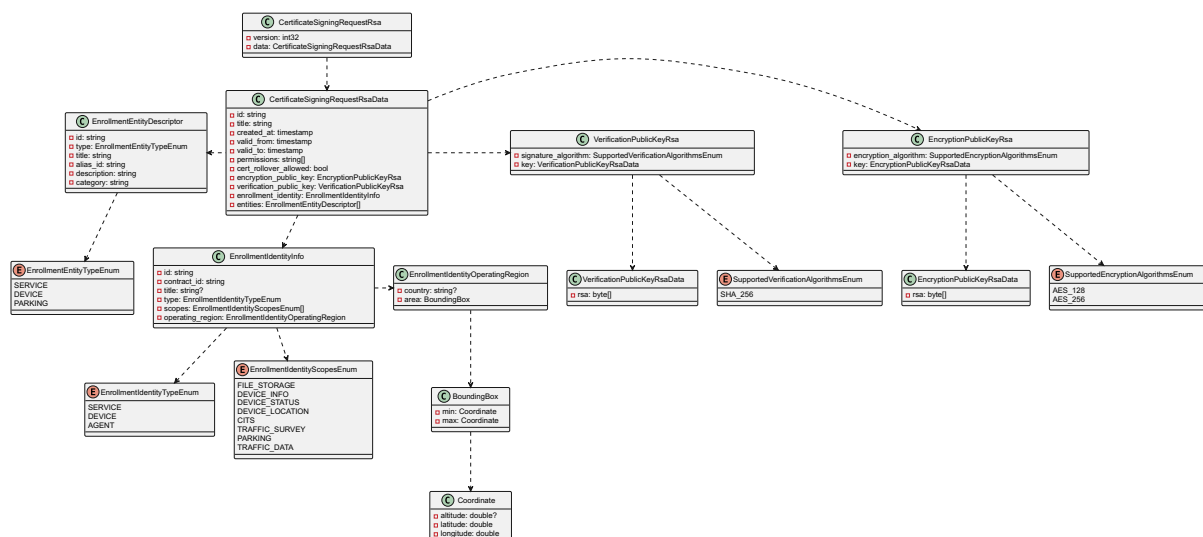
- **RSA**
- Eliptické křivky (**ECC**)

Ve formátu **RSA** je možné použít klíč o maximální síle 4096 bitů. Minimální podporovaná síla klíče **RSA** je 1024 bitů, minimální doporučená síla je 2048 bitů.

Ve formátu **ECC** je nutné použít šifru ECDS typu **NIST P.256**.

4.1.3.3 Vytvoření CSR RSA

Obsah **CSR** je nutné vyplnit do struktury objektu **CertificateSigningRequestRsa** obsaženého v datovém modelu přiložené specifikace SDGW API.



Class diagram datové struktury CSR s využití RSA šifrování

Definován bude minimálně následující výčet atributů:

CertificateSigningRequestRsa

- Version** – Verze žádosti; V případě, že jde o totožnou **EI**, která jen upravuje parametry žádosti, např. rozšiřuje scope požadovaných oprávnění, číslo je inkrementálně zvyšováno.
- Data** – Vyplněný objekt s obsahem **CSR** (**CertificateSigningRequestRsaData**)

CertificateSigningRequestRsaData

- Id** – Vygenerovaný **Guid/Uuid** žádosti
- Title** – Textový identifikátor žádosti by měl obsahovat text tak, aby byl unikátní a rozpoznatelný pro účely vyhledávání žádostí
- CreatedAt** – Datum a čas vytvoření žádost v UTC
- ValidFrom** – Datum a čas požadovaného počátku platnosti v UTC; Nelze zadat v minulosti; Nelze zadat v budoucnosti déle než **aktuální datum a čas + 90 dní**
- ValidTo** – Datum a čas požadovaného konce platnosti v UTC; Nelze zadat v minulosti; Nelze zadat delší než 12 měsíců od **ValidFrom**
- Permissions** – aktuálně nevyužívané - nevyplňovat
- EncryptionPublicKeyRsa** – Veřejná část **Encryption** klíče vygenerovaného na zařízení (**key-pair**); Klíč musí být ve formátu **PKCS #1**
- VerificationPublicKeyRsa** – Veřejná část **Verification** klíče vygenerovaného na zařízení (**key-pair**); Klíč musí být ve formátu **PKCS #1**
- EnrollmentIdentity** – Informace o **EnrollmentIdentity** (**EnrollmentIdentityInfo**)

EnrollmentIdentityInfo

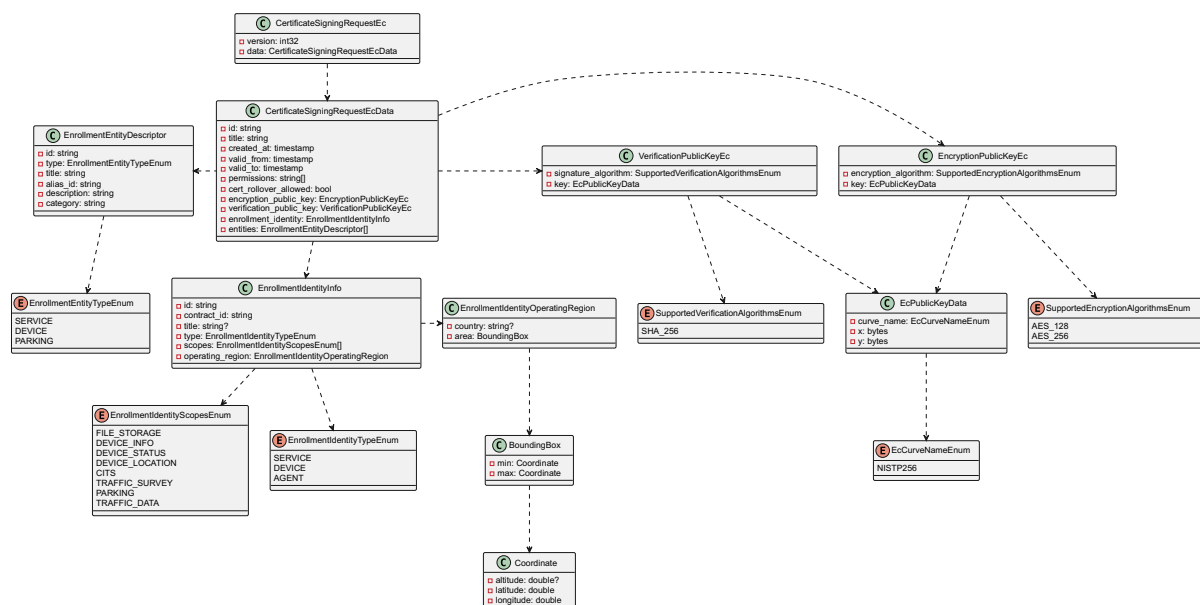
- Id** – Identifikátor **EI** zvolený integrátorem

- **ContractId** – Identifikátor smlouvy pro snazší vyhledávání v uživatelském rozhraní (nepovinné jen, pokud neexistuje)
- **Title** – Lidsky čitelný název **EI** zadaný integrátorem; Měl by reflektovat názvy, které je schopen rozpoznat a pochopit operátor centrálního systému
- **Type** – Typ **EI**
 - **NotSet** – nevalidní hodnota; **CSR** nebude systémem přijat
 - **Service** – **EI** vystupuje v roli služby integrující se systémem; Tento typ **EI** má specifická využití a nemůže vystupovat za připojená zařízení, nemůže tedy registrovat zařízení, zasílat aktualizace apod.
 - **Device** – **EI** a zařízení jsou jedna entita, tj. **EI** je integrováno přímo v zařízení
 - **Agent** – **EI** je separátní službou, která obsluhuje jednu nebo více entit (služby, zařízení)
- **Scopes** – Požadovaná oprávnění
 - **NotSet** – nevalidní hodnota; Ignorováno
 - **DeviceInfo** – Specifikace a konfigurace zařízení a jeho komponent
 - **DeviceStatus** – Stavové informace o zařízení a jeho komponentách
 - **DeviceLocation** – Polohové informace zařízení; Může být žádáno pouze **EI**, která obsluhují zařízení, která mění svou polohu; Bez tohoto **scope** je akceptována změna polohy zařízení pouze jednou za 24 hodin
 - **Cits** – Konfigurace a data související s kooperativními systémy (C-ITS)
 - **TrafficSurvey** – Konfigurace a data související se sčítači dopravy (ASD)
 - **Portal** – Konfigurace portálů
 - **Meteo** – Publikace meteorologických dat
 - **TrafficData** – Publikace dopravních dat
- **OperatingRegion** – Informace o oblasti působnosti **EI**; Využíváno jako dodatečná ochrana, kdy zařízením připojeným k takto omezené **EI** není povoleno do systému zasílat data mimo definovanou oblast působnosti; Definována musí být minimálně země (**Country**), do které je nutné hodnoty zadávat dvoupísmenný kód v souladu s **ISO 3166-1/Alpha-2**. Pro ČR je validní hodnotou **CZ**.

V okamžiku, kdy je objekt **CertificateSigningRequestRsa** vyplněn, je nutné jej serializovat do podoby binárního obsahu ve formátu **Protobuf**, a ten následně převést na **byte array**.

4.1.3.4 Vytvoření CSR ECC

Obsah **CSR** je nutné vyplnit do struktury objektu **CertificateSigningRequestEc** obsaženého v datovém modelu přiložené specifikace SDGW API.



Class diagram datové struktury CSR s využitím ECC šifrování

Definován bude minimálně následující výčet atributů:

CertificateSigningRequestEc

- Version** – Verze žádosti; V případě, že jde o totožnou **EI**, která jen upravuje parametry žádosti, např. rozšiřuje **scope** požadovaných oprávnění, číslo je inkrementálně zvyšováno.
- Data** – Vyplněný objekt s obsahem **CSR** (**CertificateSigningRequestEcData**)

CertificateSigningRequestEcData

- Id** – Vygenerovaný **Guid/Uuid** žádosti
- Title** – Textový identifikátor žádosti by měl obsahovat text tak, aby byl unikátní a rozpoznatelný pro účely vyhledávání žádostí
- CreatedAt** – Datum a čas vytvoření žádost v UTC
- ValidFrom** – Datum a čas požadovaného počátku platnosti v UTC; Nelze zadat v minulosti; Nelze zadat v budoucnosti déle než **aktuální datum a čas + 90 dní**
- ValidTo** – Datum a čas požadovaného konce platnosti v UTC; Nelze zadat v minulosti; Nelze zadat delší než 12 měsíců od **ValidFrom**
- Permissions** – aktuálně nevyužívané - nevyplňovat
- EncryptionPublicKeyEc** – Veřejná část **Encryption** klíče vygenerovaného na zařízení (**key-pair**); Klíč je definován pomocí parametrů **x**, **y** a **curvename**
- VerificationPublicKeyEc** – Veřejná části **Verification** klíče vygenerovaného na zařízení (**key-pair**); Klíč je definován pomocí parametrů **x**, **y** a **curvename**
- EnrollmentIdentity** – Informace o **EnrollmentIdentity** (**EnrollmentIdentityInfo**)

EnrollmentIdentityInfo

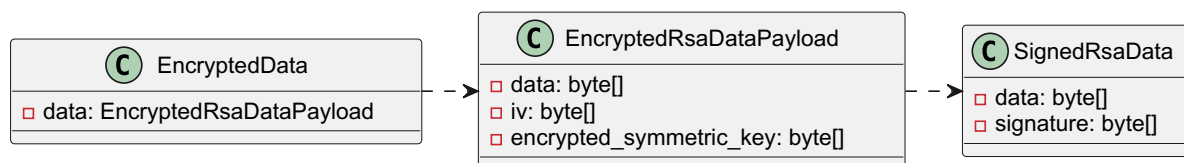
- **Id** – Identifikátor **EI** zvolený integrátorem
- **ContractId** – Identifikátor smlouvy pro snazší vyhledávání v uživatelském rozhraní (nepovinné jen, pokud neexistuje)
- **Title** – Lidsky čitelný název **EI** zadaný integrátorem; Měl by reflektovat názvy, které je schopen rozpoznat a pochopit operátor centrálního systému
- **Type** – Typ **EI**
 - **NotSet** – nevalidní hodnota; **CSR** nebude systémem přijat
 - **Service** – **EI** vystupuje v roli služby integrující se systémem; Tento typ **EI** má specifická využití a nemůže vystupovat za připojená zařízení, nemůže tedy registrovat zařízení, zasílat aktualizace apod.
 - **Device** – **EI** a zařízení jsou jedna entita, tj. **EI** je integrováno přímo v zařízení
 - **Agent** – **EI** je separátní službou, která obsluhuje jednu nebo více entit (služby, zařízení)
- **Scopes** – Požadovaná oprávnění
 - **NotSet** – nevalidní hodnota; Ignorováno
 - **DeviceInfo** – Specifikace a konfigurace zařízení a jeho komponent
 - **DeviceStatus** – Stavové informace o zařízení a jeho komponentách
 - **DeviceLocation** – Polohové informace zařízení; Může být žádáno pouze **EI**, která obsluhují zařízení, která mění svou polohu; Bez tohoto **scope** je akceptována změna polohy zařízení pouze jednou za 24 hodin
 - **Cits** – Konfigurace a data související s kooperativními systémy (C-ITS)
 - **TrafficSurvey** – Konfigurace a data související se sčítači dopravy (ASD)
 - **Portal** – Konfigurace portálů
 - **Meteo** – Publikace meteorologických dat
 - **TrafficData** – Publikace dopravních dat
- **OperatingRegion** – Informace o oblasti působnosti **EI**; Využíváno jako dodatečná ochrana, kdy zařízením připojeným k takto omezené **EI** není povoleno do systému zasílat data mimo definovanou oblast působnosti; Definována musí být minimálně země (**Country**), do které je nutné hodnoty zadávat dvoupísmenný kód v souladu s **ISO 3166-1/Alpha-2**. Pro ČR je validní hodnotou **CZ**.

V okamžiku, kdy je objekt **CertificateSigningRequestEc** vyplněn, je nutné jej serializovat do podoby binárního obsahu ve formátu **Protobuf**, a ten následně převést na **byte array**.

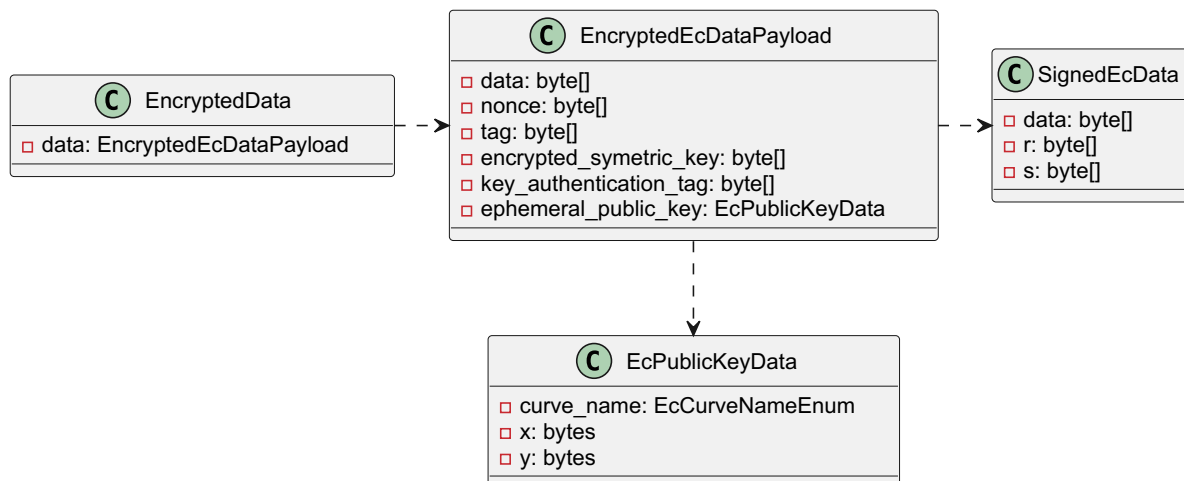
4.1.3.5 Podpis, šifrování a odeslání CSR do služby Enrollment Authority

Před odesláním do rozhraní SDGW je obsah žádosti o enrollment (**Enrollment Request**) potřeba podepsat a zašifrovat pomocí **Encryption** key získaného ze tohoto rozhraní.

Následující diagram zobrazuje strukturu datových modelů pro RSA i ECC typ šifrování. Šipky definují závislosti, nikoliv směr toku dat (ten je z pohledu **EI** opačný).



Struktura datového modelu pro RSA šifrování



Struktura datového modelu pro ECC šifrování

Veřejné části klíčů vyměňované mezi zařízením a serverem jsou vždy ve formátu **PKCS #1** (pouze pro RSA) (https://en.wikipedia.org/wiki/PKCS_1). Pro metodu eliptických křivek je naopak využito zaslání parametrů **x**, **y** a **curve_name**.

4.1.3.6 Generování digitálního podpisu obsahu CSR pro metodu RSA

Ze serializovaného **CSR** uložené v podobě **byte array** je nutné spočítat hash pomocí **SHA256**. Tento hash je nutné podepsat privátním verifikačním klíčem **EI** (**Verification key-pair**) a výstup převést do **byte array** ve formátu **PKCS #1**.

Obě hodnoty (serializovaný **CSR** a podepsaný hash) je nutné vyplnit do objektu **SignedRsaData** kde je do property **Data** vložen serializovaný obsah **CSR** objektu, a do **Signature** je vložen vytvořený podpis (podepsaný **SHA256** hash obsahu).

4.1.3.6.1 Šifrování obsahu CSR pomocí RSA určeného pro službu EA

Před šifrováním obsahu je nejprve nutné na **EI** vygenerovat **IV** (**Initialization Vector**) a symetrický klíč (**Symmetric Key**). Následně je vyplněný objekt **SignedRsaData** jako celek serializován pomocí **Protobuf** do podoby **byte array**, a s využitím **IV** a symetrického klíče jsou tato data zašifrována metodou **AES-CBC** s paddingem ve standardu dle **PKCS#7**.

Objekt **EncryptedRsaData** je naplněn zašifrovaným obsahem **CSR** (property **Data**), a dále je vyplněna hodnota **IV**.

V tuto chvíli je ještě potřeba zašifrovat symetrický klíč – tento je potřeba zašifrovat pomocí `Encryption Public Key` získaného z rozhraní SDGW. Tuto hodnotu je nutné převést do podoby `byte array` a tu vyplnit do zbývajících property objektu `EncryptedRsaData` s názvem `EncryptedSymmetricKey`.

Tento objekt je nutné opět serializovat pomocí `Protobuf` do podoby `byte array`.

4.1.3.6.2 Šifrování obsahu CSR pomocí ECC určeného pro službu EA

Před šifrováním obsahu je nejprve nutné na `EI` vygenerovat:

- `Nonce`
- `Tag`
- `Symetric key`
- `Key_auth tag`
- `Ephemeral public key`

Způsob generování hodnot vychází z algoritmu `ECIES`. Popis generování výše zmíněných parametrů je možné nalézt ve standardu `IEEE 1609.2`.

4.1.3.6.3 Odeslání připraveného CSR do služby EA

Takto podepsaná, zašifrovaná, a nakonec serializovaná, data lze vyplnit do property `EncryptedRsaData` či `EncryptedEcData` objektu `EncryptedData` a s využitím metody `EnrollIdentityRsa()` nebo `EnrollIdentityEc()` odeslat požadavek do služby `Enrollment Authority` ke schválení.

4.1.3.6.4 Pseudokód pro RSA šifrování

Protože každá aplikace, služba a zařízení je vyvíjena pomocí odlišných programovacích jazyků a nástrojů, následující pseudokód může usnadnit pochopení procesu podpisu a šifrování.

```
// RSA encryption flow

function EncryptData(data, EaServiceRsaPublicEncryptionKey)
{
    EaServiceRsaPublicEncryptionKey = call GetEncryptionPublicKey()

    byte[] aesKey = GenerateAesKey()

    byte[] iv = GenerateAesIv()

    byte[] aesEncryptedData = AesCbcEncrypt(aesKey, iv, data) // AES CBC mode with PKCS#7 padding!

    byte[] encryptedSymmetricKey = RsaEncrypt(EaServiceRsaPublicEncryptionKey, aesKey)

    EncryptedRsaData encryptedRsaData = new EncryptedRsaData
    {
        Data = new EncryptedRsaDataPayload
        {
            IV = iv,
            Data = aesEncryptedData,
            EncryptedSymmetricKey = encryptedSymmetricKey
        }
    }
}
```



```

    },
}

return encryptedRsaData
}

// RSA Signature flow

function SignCsr(data, csrPrivateVerificationKey)
{
    byte[] signature = RsaSign(csrPrivateVerificationKey, data, HashAlgorithmName.Sha256, RsaSignaturePadding.Pkcs1)

    SignedRsaData signedData = new SignedRsaData
    {
        SignedData = new SignedRsaDataPayload
        {
            Data = data,
            Signature = signature
        },
        VerificationPublicKey = new VerificationPublicKeyRsa
        {
            SignatureAlgorithm = SupportedVerificationAlgorithms.Sha256,
            Key = new VerificationPublicKeyRsaData
            {
                Rsa = verificationPublicKey
            },
        },
    },
}

return signedData
}

// Process CSR

CsrRequestId csrId = new Guid()

// Gather devices required to be preregistered along with Enrollment Request
EnrollmentEntityIdentifier[] entities = GetAllDevicesAsEnrollmentIdentifiers()

CertificateSigningRequestRsa csr = GenerateCsrData(csrId, entities)

byte[] csrContent = SerializeCsrContentToProtobuf(csr)

SignedRsaData signedData = SignCsr(csrContent, csrPrivateVerificationKey)

byte[] signedCsrContent = SerializeSignedDataToProtobuf(signedData)

EncryptedRsaData encryptedRsaData = EncryptData(signedCsrContent, EaServiceRsaPublicEncryptionKey)

// Send to Enrollment Authority service

call EnrollIdentityRsa(new EnrollIdentityRequest
{
    Id = csrId,
    CsrEncryptedData = EncryptedData
})

```

4.1.3.6.5 Pseudokód pro EC šifrování

Protože každá aplikace, služba a zařízení je vyvíjena pomocí odlišných programovacích jazyků a nástrojů, následující pseudokód může usnadnit pochopení procesu podpisu a šifrování.

```
// EC encryption flow

function EncryptData(data, EaServiceEcPublicEncryptionKey)
{
    EaServiceEcPublicEncryptionKey = call GetEncryptionPublicKey()
    byte[] aesKey = GenerateAesKey()
    byte[] nonce = GenerateNonce()
    var encryptedEncryptionKey = EncryptAesKey(aesEncryptionKey, encryptionPublicKey.Key);
    using var aesCcm = new AesCcm(aesEncryptionKey);
    var cipherText = new byte[plainTextData.Data.Length];
    var tag = new byte[AesKeyLength];
    aesCcm.Encrypt(nonce, plainTextData.Data, cipherText, tag);
    var encryptedEcData = new EncryptedEcData
    {
        EncryptedSymmetricKey = encryptedEncryptionKey.EncryptedSymmetricKey,
        KeyAuthenticationTag = encryptedEncryptionKey.KeyAuthenticationTag,
        EphemeralPublicKey = encryptedEncryptionKey.EphemeralPublicKey,
        Nonce = nonce,
        Data = cipherText,
        Tag = tag,
    }
    return encryptedEcData;
}

function EncryptAesKey(byte[] aesCcmKey, EcPublicKeyData recipientPublicKey)
{
    var keyPair = GenerateEphemeralKeyPair();
    var agreement = new ECDHBasicAgreement();
    var recipientPublicKeyParameters = CreateBcPublicKeyParameters(recipientPublicKey);
    agreement.Init(keyPair);
    var ss = agreement.CalculateAgreement(recipientPublicKey);
    var ssArray = BigIntegers.AsUnsignedByteArray(agreement.GetFieldSize(), ss);
    var k1K2 = new byte[aesLength + length];
    var kdf2BytesGenerator = new Kdf2BytesGenerator(new Sha256Digest());
    kdf2BytesGenerator.Init(new KdfParameters(ssArray, Array.Empty<byte>()));
    kdf2BytesGenerator.GenerateBytes(k1K2, 0, k1K2.Length);
    var c = new byte[aesLength];
    for (var i = 0; i < aesLength; i++)
        c[i] = (byte)(aesCcmKey[i] ^ k1K2[i]);
    var mac = new HMac(new Sha256Digest());
    mac.Init(new KeyParameter(k1K2, aesLength, length));
    foreach (var val in c)
        mac.Update(val);
    var macResult = new byte[mac.GetMacSize()];
    mac.DoFinal(macResult, 0);
    var t = new byte[aesLength];
    Array.Copy(macResult, t, aesLength);
    var ephemeralPublicKeyData = new EcPublicKeyData
    {
        CurveName = recipientPublicKey.CurveName,
        X = keyPair.publicKeyParameters.Q.AffineXCoord.GetEncoded(),
        Y = keyPair.publicKeyParameters.Q.AffineYCoord.GetEncoded(),
    };
    return new EncryptedEncryptionKey(c, t, ephemeralPublicKeyData);
}

// EC encryption flow

function SignCsr(data, csrPrivateVerificationKey)
{
    (byte[] r, byte[] s) = EcdsaSign(csrPrivateVerificationKey, data, HashAlgorithmName.Sha256)
    SignedEcData signedData = new SignedEcData
    {
        SignedData = new SignedEcDataPayload
        {
            Data = data,
            R = r,
            S = s,
        },
        VerificationPublicKey = new EcVerificationPublicKey
    }
}
```

```

        SignatureAlgorithm = SupportedVerificationAlgorithms.Sha256,
        Key = new EcPublicKeyData
        {
            CurveName = EcCurveName.NistP256,
            X = publicKey.Q.X,
            Y = publicKey.Q.Y
        },
    },
}
return signedData
}

// Process CSR

CsrRequestId csrId = new Guid()

// Gather devices required to be preregistered along with Enrollment Request

EnrollmentEntityIdentifier[] entities = GetAllDevicesAsEnrollmentIdentifiers()
CertificateSigningRequestEc csr = GenerateCsrData(csrId, entities)
byte[] csrContent = SerializeCsrContentToProtobuf(csr)
SignedEcdData signedData = SignCsr(csrContent, csrPrivateVerificationKey)
byte[] signedCsrContent = SerializeSignedDataToProtobuf(signedData)
EncryptedEcData encryptedEcData = EncryptData(signedCsrContent, EaServiceEcPublicEncryptionKey)

// Send to Enrollment Authority service

call EnrollIdentityEc(new EnrollIdentityEcRequest)
{
    Id = csrId,
    CsrEncryptedData = encryptedEcData
}

```

4.1.3.7 Čekání na odpověď ze služby Enrollment Authority

Metody `EnrollIdentityRsa()` a `EnrollIdentityEc()` jsou gRPC metody typu `Server Streaming`, což fakticky znamená, že voláním těchto metod je otevřeno persistentní spojení s rozhraním SDGW, kdy klient (EI) čeká na odpověď serveru s vygenerovaným certifikátem.

Obě metody mají následující vstupní parametry

- `Id` – Identifikátor požadavku datového typu `Guid` – je shodný s `Id` požadavku uvedeným v objektu `CertificateSigningRequestData`
- `CsrEncryptedData` – podepsaný a zašifrovaný obsah `CSR`

Po odeslání `CSR` ponechá služba EI spojení otevřeno a čeká na příjem odpovědi. EI ukončí spojení v okamžiku odpovědi indikující úspěch.

Po úspěšně získané odpovědi s vygenerovaným certifikátem následně klient (EI)

- Odstraní vygenerovaný `CSR`
- Persistentně uloží získaný certifikát na bezpečné místo

Odpověď serveru může být v některých případech závislá na akci uživatele. Nemusí tak dorazit dny, týdny ani měsíce. EI je povinná být stále připojena a čekat na výsledek požadavku.

4.1.3.8 Schválení EI a vygenerování certifikátu

Od okamžiku schválení (vydání certifikátu) rozhraním SDGW je **Enrollment Identity**, tedy zařízení nebo jiný systém autorizovaný pro komunikaci se centrálním systémem, oprávněn:

- a. Žádat rozhraní SDGW o vydání **auth** tokenu
- b. Registrovat, prostřednictvím rozhraní SDGW, entity do centrálního systému
- c. Aktualizovat provozní a stavová data takových entit
- d. Publikovat dopravní a jiná telematická data (meteorologická data, sčítací data, detekce apod.)

4.1.3.9 Řešení potíží

Integrátor musí zajistit, že aplikace, služby a zařízení systém neúměrně nezatěžují. V případě přetěžování je takové chování hodnoceno jako defekt dodávky a integrátor je povinen provést odstranění závady. Do doby odstranění závady může být aplikace, služba nebo zařízení odstaveno z provozu a po dobu odstávky může být ze strany centrálního systému považováno za nefunkční, tj. totožně jako kdyby nekomunikovalo, a to se všemi důsledky vyplývajícími ze smluvního vztahu. Tato aktivita je monitorována a akceptovaná míra je maximálně 5 incidentů za posledních 365 dní.

V procesu integrace může dojít k následujícím potížím.

4.1.3.9.1 Ztráta spojení

Při ztrátě spojení s rozhraním SDGW, ať už v důsledku výpadku konektivity nebo restartu aplikace, služby nebo zařízení, je možné již jednou vygenerovaný **CSR** zaslat v požadavku znovu.

4.1.3.9.2 Vygenerování nové CSR

V případě, že vygenerované **CSR** není k dispozici, je nutné **CSR** vygenerovat, podepsat a zašifrovat znovu. Znamená to však novou žádost v systému.

Z důvodu zachování efektivity by mělo jít pouze o výjimečný proces – tj. danou operaci (generování stále nového CSR) nelze provádět při každém startu aplikace, služby nebo zařízení.

4.1.3.9.3 Ztráta vydaného certifikátu

V případě ztráty vydaného certifikátu je možné pomocí rozhraní SDGW zaslat žádost o nový certifikát. Původní certifikát bude přidán na seznam revokovaných certifikátů.

Z důvodu zachování efektivity by mělo jít pouze o výjimečný proces – tj. danou operaci nelze provádět při každém startu aplikace, služby nebo zařízení.

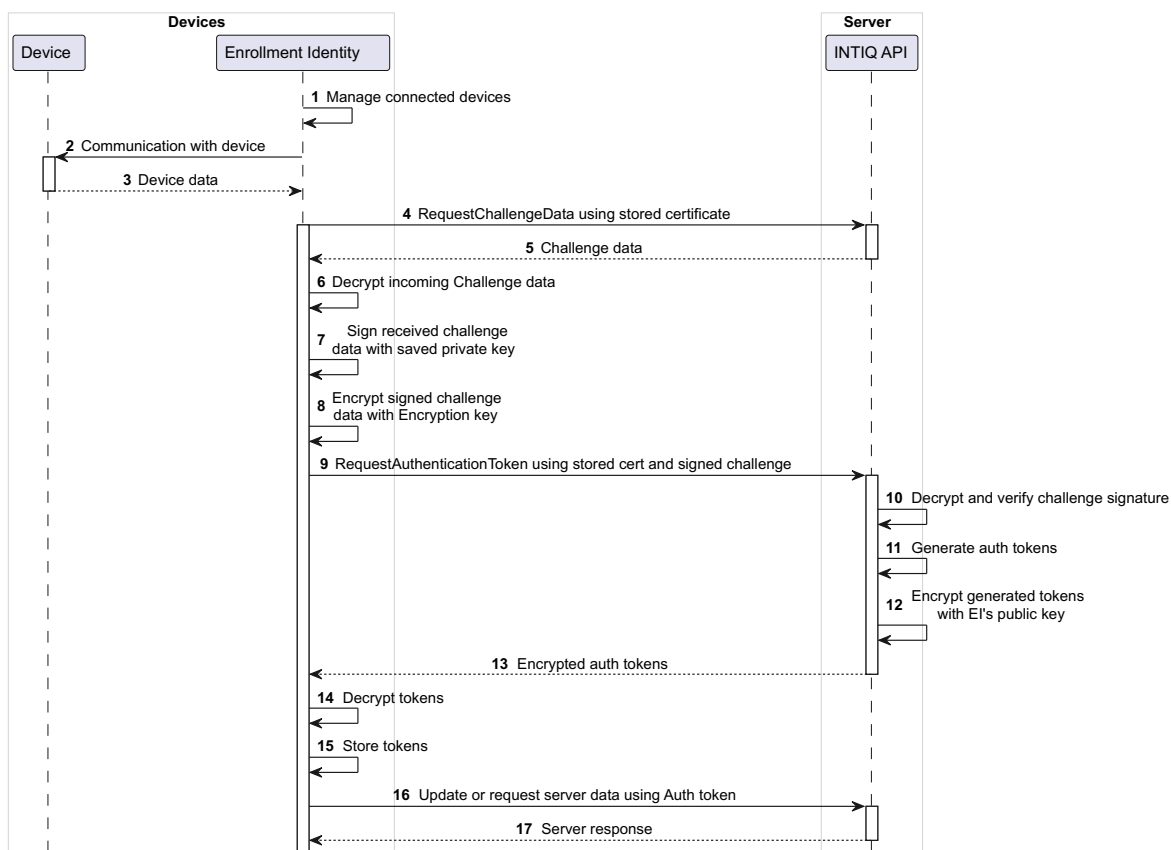
4.1.3.9.4 Ztráta privátního klíče aplikace, služby nebo zařízení

Pokud je **key-pair** v rámci **Enrollment Identity**, která již byla v systému Enrollment Authority schválena, a pro kterou již byl vydán certifikát, znovu vygenerován, bude taková **EI** považována za neautorizovanou a bude nutné znovu provést proces enrollmentu.

4.2 Autentizace a správa tokenů

V předchozí části dokumentu (5.1 Autorizace Enrollment Identity) bylo uvedeno, jak **Enrollment Identity** enrollovat do centrálního systému a získat certifikát. Nicméně samotný certifikát entitě neopravňuje ke komunikaci s rozhraním SDGW. K tomu je potřeba získat **auth** token, který je využíván k autentizaci i autorizaci požadavků zasílaných na toto rozhraní.

Tato část dokumentu popisuje, jak **auth** token získat, jak s ním nakládat a jak jej použít k autentizaci a autorizaci vůči rozhraní SDGW.



Sekvenční diagram získání auth tokenu

4.2.1 Autorizované identifikátory

Rozhraní SDGW umožní službě, zařízení nebo jiné **EI** komunikovat za jiné zařízení pouze v případě, že je identifikátor této entity obsažen v **auth** tokenu, který je přiložen v requestu na rozhraní SDGW. Rozhraní SDGW odmítne předat nebo aktualizovat data entity, která v takovém tokenu není uvedena.

Aby entity (zařízení) s konkrétním identifikátorem byly uvedeny jako součást `auth` tokenu, je nutné uvést jeden nebo více takových identifikátorů v žádosti o vydání `auth` tokenu.

Výčet identifikátorů je v `auth` tokenu složen z typu entity (`EnrollmentEntityIdentifierType`) a faktického identifikátoru zvoleného na straně `EI`. Identifikátor musí být v rámci jedné `EI` unikátní. Kolize identifikátorů jsou v rámci rozhraní SDGW přípustné, pokud jsou registrovány odlišnými `EI`.

Rozhraní SDGW odlišuje následující typy identifikátorů

Identifikátor	Popis
ENROLLMENT_ENTITY_TYPE_ENUM_DEVICE	Zařízení
ENROLLMENT_ENTITY_TYPE_ENUM_SERVICE	Služba
ENROLLMENT_ENTITY_TYPE_ENUM_PARKING	Parkovací lokalita

4.2.1.1 Specifikace identifikátorů zařízení

Řídící zařízení nebo jiná `EI` může v jedné žádosti žádat pouze o omezený počet identifikátorů – aktuální limit je stanoven na 50 souhrnně za všechny entity. V praxi to znamená, že pokud `EI` obsluhuje více entit než tento limit, musí pracovat s více aktivními `auth` tokeny současně. Pokud je v žádosti více identifikátorů, je takový požadavek odmítnut a zalogován.

Minimální počet identifikátorů je dán celkovým počtem aktivních identifikátorů registrovaných `EI` v rozhraní SDGW. `EI` v každé žádosti požaduje maximální možný počet identifikátorů, tj. nežádá o jednotlivé tokeny pro jednotlivé entity.

V praxi to znamená, že pokud řídící zařízení obsluhuje více entit než výše uvedený limit, musí pracovat s více aktivními `auth` tokeny současně. Pokud `EI` registruje např. 102 identifikátorů, je nutné si identifikátory rozdělit do skupin po 50 entitách. Výsledkem jsou tři žádosti, z toho tři žádosti mají plný počet 50 identifikátorů, jedna žádost pak obsahuje pouze dva identifikátory.

V případě jedné služby komunikující za 102 zařízení, je nezbytné, aby služba v jedné chvíli udržoval 3 aktivní `auth` tokeny

- první token a druhý token obsahující vždy 50 zařízení
- třetí token obsahuje na zbylá 2 zařízení

Služba (nebo jiná `EI`) musí při žádosti o `auth` token s identifikátory entit respektovat následující pravidla

- `EI` může žádat pouze takové identifikátory entit, které sama předtím registrovala
- `EI` musí mít žádat jen o identifikátory entit, které byly schváleny a jsou vedeny jako aktivní
- Maximálně 50 identifikátorů v jednom požadavku

Identifikátory je v objektu `AuthTokenRequestData` žádosti o vydání `auth` tokenu vyplňovat do property s názvem `entities`. V žádosti o token postačí vyplnění pouze typu entity (`type`) a jejího identifikátoru (`id`). Výčet identifikátorů v jednom tokenu lze skládat z různých typů.

`EI` musí počítat s tím, že rozhraní SDGW může vydání `auth` tokenu odmítnout nebo vrátit token, který neobsahuje všechny žádané identifikátory.

Pokud je žádost úspěšná, rozhraní SDGW vrátí `auth` token společně se seznamem entit, pro které byla žádost o token rozhraním akceptována.

4.2.2 Challenge data

Každá `EI` v enrollovaném stavu musí rozhraní SDGW pravidelně prokazovat, že stále vlastní privátní klíče (`Verification` a `Encryption`), na jejichž základě byla integrace s rozhraním SDGW schválena.

Rozhraní SDGW za tímto účelem vynucuje omezení platnosti vydávaných `auth` tokenů, kdy při každé následné žádosti o nový token musí `EI` znovu a znovu prokazovat vlastnictví podpisem kontrolních dat získaných z rozhraní SDGW (`ChallengeData`).

`ChallengeData` jsou z rozhraní SDGW získána pomocí volání `GetChallengeDataRsa()` nebo `GetChallengeDataEc()` – záleží na použitém typu šifrování. Jako součást požadavku je nutné zaslat následující HTTP hlavičky

- `X-HTTP-CERT` – certifikát získaný z rozhraní SDGW po schválení `CSR` převedený do podoby `BASE64` encoded string.

V případě úspěchu volání jsou data vrácena v zašifrované podobě. Tato data je nutné dešifrovat pomocí vlastního privátního `Encryption` key. Proces dešifrování je ekvivalentní procesu šifrování, s následujícími rozdíly

- Proces je zrcadlově obrácený
- Využívány jsou vždy opačné klíče (`private` → `public`; `public` → `private`) a opačných stran – tam kde byl využíván veřejný `Encryption` key protistrany je nyní využit privátní `Encryption` key `EI`.

V okamžiku, kdy má `EI` k dispozici dešifrovaná `ChallengeData`, tedy `Id` a `Data` v podobě `byte array`, může rozhraní SDGW požádat o vygenerování `auth` tokenu.

4.2.3 Získání auth tokenu

`Auth` token lze z rozhraní SDGW získat voláním funkce `GetAuthTokenRsa()` nebo `GetAuthTokenEc()`. Jako součást požadavku je nutné zaslat následující HTTP hlavičky

- `X-HTTP-CERT` – certifikát získaný z rozhraní SDGW po schválení `CSR` převedený do podoby `BASE64` encoded string.

Pro úspěšné volání této metody je dále nutné, totožným způsobem jako v případě `CSR`, podepsat a zašifrovat vyplněný objekt `AuthTokenRequestData`. Pro tento objekt je potřeba zajistit následující data:

- `Id` – Identifikátor kontrolních dat vrácené z metod `GetChallengeDataRsa()` / `GetChallengeDataEc()`
- `ChallengeData` – dešifrovaná kontrolní data vrácená z metod `GetChallengeDataRsa()` / `GetChallengeDataEc()`
- `Entities` – výčet identifikátorů entit (zařízení), pro která má být požadovaný token vydán – max. 50
- `Scopes` – omezení scopů v maximálním rozsahu položek schválených pro tuto `Enrollment Identity`; Vhodné zejména pro operace, které využívají jen některé vybrané funkce rozhraní SDGW

V případě úspěchu server zašle podepsaná a zašifrovaná data, která je nutné deserializovat pomocí `Protobuf` do objektu `AuthTokens`, která poskytuje dva typy tokenů

- `AccessToken` – autentizační a autorizační token (`auth` token) pro komunikaci s ostatními službami centrálního systému; kratší platnost; Na základě `AccessToken` nelze vyžádat nový token
- `RefreshToken` – token pro možnost vyžádání nového `AccessToken` bez nutnosti získávat a podepisovat `ChallengeData` apod.; Nelze použít pro komunikaci se službami; Nelze vyžádat nový `RefreshToken`
- `AuthorizedEntities` – výčet autorizovaných zařízení pro vydaný token; V seznamu jsou pouze zařízení, která byla schválena do provozu Zadavatelem – tj. neobsahuje ještě neschválená zařízení

Získané tokeny (`Access` a `Refresh`) jsou v `EI` drženy pouze v paměti zařízení. Tyto tokeny nejsou v žádné aplikaci, službě či zařízení persistentně ukládány.

4.2.4 Obnova auth tokenu

Obnovu `auth` tokenu je zpravidla nutné zajistit ve 2/3 platnosti aktuálně platného `auth` tokenu. V tuto chvíli rozhraní SDGW podporuje následující způsoby získání `auth` tokenů.

4.2.4.1 Obnova pomocí Refresh token

Pokud má `EI` k dispozici platný `Refresh` token, je obnovu `auth` tokenu možné provést voláním `RefreshAuthTokenRsa()` nebo `RefreshAuthTokenEc()`, kterému je `Refresh` token předán v zašifrované podobě. Šifrování je totožné jako v případě `CSR`.

4.2.4.2 Žádost o nový token

Pokud `EI` nemá platný `Refresh` token, musí `EI` žádat o nový `auth` token voláním metody `GetAuthTokenRsa()` nebo `GetAuthTokenEc()` – viz výše.

4.2.4.3 Implementační omezení

Řídicí zařízení (nebo jiná `EI`) je povinna implementovat mechanismus obnovy (získání `Access` token) s využitím `Refresh` tokenů. O vydání nového `Access` tokenu pomocí `Refresh` tokenu je `EI` povinna žádat rozhraní SDGW nejdříve po uplynutí 2/3 doby mezi datem a časem vydáním a koncem platnosti `Access` tokenu.

Pokud `EI` nemá k dispozici platný `Refresh` token, je nutné žádat o nový pár tokenů (`Access` a `Refresh`). `EI` v takovém případě nesmí rozhraní SDGW neúměrně zatěžovat. Žádost o vydání nového páru `auth` tokenů (metoda `GetAuthTokenRsa()` nebo `GetAuthTokenEc()`) může `EI` žádat nejdříve po uplynutí 8/10 času platnosti od získání aktuálně evidovaného platného `Refresh` tokenu.

V případě nefunkčnosti výše uvedeného procesu je takové chování hodnoceno jako defekt dodávky a Dodavatel je povinen provést odstranění závady. Do doby odstranění závady může být `EI` nebo její část odstavena z provozu. Po dobu odstávky může být ze strany Zadavatele považována zařízení za nefunkční, tj. totožně jako kdyby nekomunikovala, a to se všemi důsledky vyplývajícími ze smluvního vztahu (SLA). Tato aktivita je monitorována a akceptovatelná míra incidence je maximálně 20 incidentů porušení nakládání s `auth` tokeny za posledních 365 dní.

4.2.5 Využití auth tokenů k autentizaci a autorizaci

Platný **Access** token je možné využít k autentizaci volání služeb rozhraní SDGW. Autentizace a autorizace volání je zajištěna v případě přidání tohoto tokenu do HTTP hlavičky **Authorization**, včetně textu **Bearer**.

Příklad hodnoty předávané v hlavičce

```
Authorization: Bearer eyJhbGciOiJIUzI1Ni.IXVCJ9TJVr7E20RMHrHDcEfxj.oYZgeFONFh7HgQ
```

4.2.5.1 Autorizace volání pro konkrétní entitu

Každý **auth** token opravňuje držitele ke komunikaci pro předem definovaný výčet entit (zařízení), která **EI** registrovala a následně uvedla v žádosti o token. S daným tokenem je tedy možné provést volání API (**unární**) nebo otevřít kanál (**streaming**) jen pro práci s daty právě takového výčtu zařízení.

Pro potvrzení toho, která zařízení jsou (oproti žádosti) pro takový token autorizována, je nutné vyhodnotit parametr **AuthorizedEntities**, vrácený s odpovědí na žádost o **auth** token.

V případě potřeby práce s daty většího množství zařízení, než jaké stanoví limit (viz 5.2.1.1 Specifikace identifikátorů zařízení), je potřeba, aby **EI** žádala o více tokenů najednou. Na základě takových tokenů pak bude provádět aktualizace ve skupinách.

4.2.5.2 Ošetření návratových stavů

Server může vrátit následující stavy požadavků:

- **Unauthorized** – **auth** token není platný; Je potřeba obnovit token nebo získat nový
- **Forbidden** nebo **Not Found** – klient nemá dostatečná oprávnění
- **Jiný** – nesouvisí s autorizací

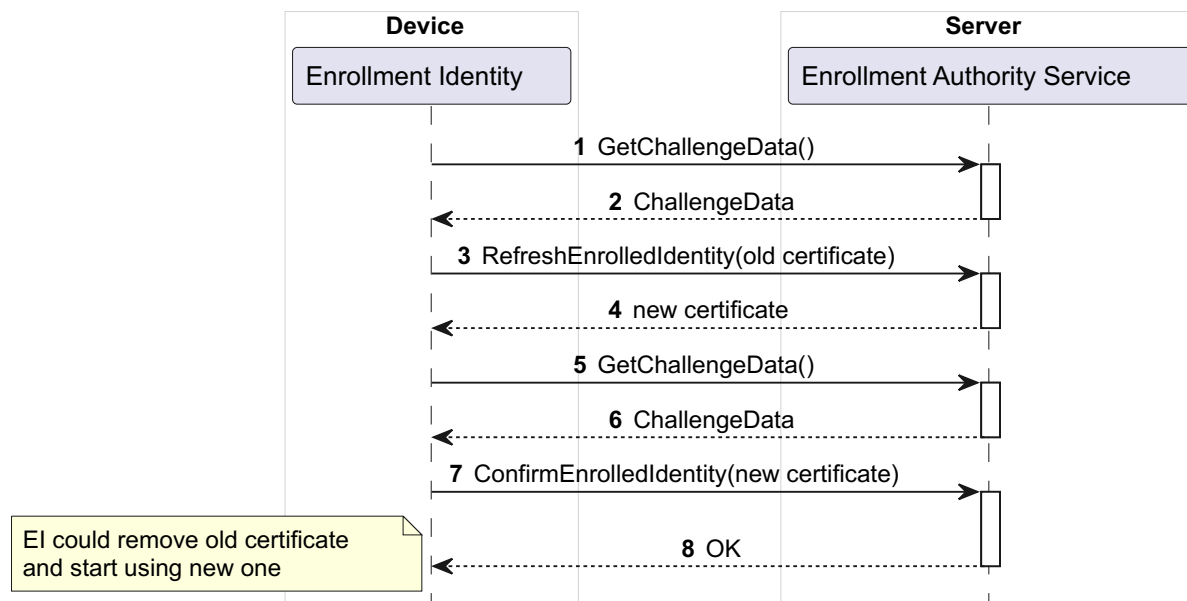
4.2.6 Obnova certifikátu

Získaný certifikát má vždy omezenou platnost (např. 12 měsíců). V přibližně jeho 2/3 platnosti je nutné tento certifikát obnovit pomocí metod **RefreshEnrolledIdentityRsa()** / **RefreshEnrolledIdentityEc()** a **ConfirmEnrolledIdentityRsa()** / **ConfirmEnrolledIdentityEc()**.

Metoda **RefreshEnrolledIdentity*()** slouží k získání nového certifikátu. V **requestu** metody je zaslán původní certifikát. Server následně vrátí nový certifikát. Tento nově vydaný certifikát je však možné používat až po uložení nového certifikátu klientem (**EI**) a potvrzení klientem (**EI**) serveru (**EA**).

Protože v jednom okamžiku je možné pro komunikaci se serverem používat pouze jeden certifikát, není, do doby konfirmace nového certifikátu, možné se serveru tímto certifikátem prokazovat pro běžnou komunikaci (vyžádání tokenu apod.) Komunikace prostřednictvím nepotvrzeného certifikátu je možná pouze za účelem vydání a potvrzení nového.

Konfirmaci je možné potvrdit voláním metody **ConfirmEnrolledIdentity*()**, kdy je nutné zaslat nový (přijatý) certifikát a **ChallengeData**, která je nutné podepsat původním privátním klíčem. **ChallengeData** nesmí být podepsány novým privátním klíčem. Takový požadavek je považován za neplatný. Po úspěšném potvrzení přijetí požadavku je původní certifikát na straně serveru zneplatněn a od té chvíle je možné pro komunikaci se serverem používat nový certifikát.



Zjednodušený sekvenční diagram obnovy certifikátu

4.2.7 Řešení potíží

4.2.7.1 Vypršení platnosti Refresh tokenu

Při vypršení platnosti **Refresh** tokenu není možné na jeho základě žádat o nový **Access** token. V tomto případě je nutné žádat o vydání nového páru **Access** a **Refresh** tokenu.

4.2.7.2 Ztráta auth tokenu

Při ztrátě tokenu, např. z důvodu restartu zařízení, je nutné žádat o vydání nového páru **Access** a **Refresh** tokenu.

4.2.7.3 Odmítání vydání auth tokenu

Pokud rozhraní SDGW odmítá žádosti o vydání nového **auth** tokenu, ale přesto se zdá být vše ostatní v pořádku (např. služba **EI** je úspěšně enrolována do systému a systém nevykazuje konkrétní nedostatky), je nutné kontaktovat správce systému.

4.2.8 Seznam dostupných metod služby EA

Pro implementaci výše uvedených funkcí služby Enrollment Authority jsou k dispozici následující metody.

Metoda	Typ	Vyžadovaný Scope	Popis
--------	-----	------------------	-------

<i>GetEncryptionPublicKeyRsa</i>	unární	-	Získání veřejné části Encryption klíče za účelem šifrování dat směřujících z klienta do služby <i>EA</i>
<i>GetEncryptionPublicKeyEc</i>			
<i>EnrollIdentityRsa</i>	Server	-	Požadavek na enrollment služby <i>EI</i> do rozhraní SDGW. Výsledkem je získání certifikátu, který je následně používán pro další komunikaci se službou <i>EA</i> .
<i>EnrollIdentityEc</i>	streaming		
<i>RefreshEnrolledIdentityRsa</i>	unární	-	Obnova již enrolované identity validním certifikátem. Metoda vrací nový certifikát, který je nutné potvrdit pomocí metod <i>ConfirmEnrolledIdentity</i>
<i>RefreshEnrolledIdentityEc</i>			
<i>ConfirmEnrolledIdentityRsa</i>	unární	-	Potvrzení nového certifikátu <i>EI</i>
<i>ConfirmEnrolledIdentityEc</i>			
<i>GetChallengeDataRsa</i>	unární	-	Získání kontrolního obsahu ze služby <i>EA</i> za účelem prokázání vlastnictví privátního klíče na straně <i>EI</i>
<i>GetChallengeDataEc</i>			
<i>GetAuthTokenRsa</i>	unární	-	Žádost o <i>auth</i> token (<i>Access</i> a <i>Refresh</i>) ze služby <i>EA</i> . Pro získání tokenu je nutné vlastnit certifikát a mít podepsány kontrolní data pro ověření vlastnictví privátního klíče (<i>ChallengeData</i>)
<i>GetAuthTokenEc</i>			
<i>RefreshAuthTokenRsa</i>	unární	-	Žádost o nový <i>Access</i> token na základě platného <i>Refresh</i> tokenu.
<i>RefreshAuthTokenEc</i>			

4.2.9 Oprávnění

Při žádosti o enrollment je možné definovat požadovaná oprávnění **Enrollment Identity** prostřednictvím parametru **Scopes**. Oprávnění lze nastavit na následující položky

- **DeviceInfo** – Specifikace a konfigurace zařízení a jeho komponent
- **DeviceStatus** – Stavové informace o zařízení a jeho komponentách
- **DeviceLocation** – Polohové informace zařízení; Může být žádáno pouze **EI**, která obsluhují zařízení, která mění svou polohu; Bez tohoto **scope** je akceptována změna polohy zařízení pouze jednou za 24 hodin
- **Cits** – Konfigurace a data související s kooperativními systémy (C-ITS)
- **TrafficSurvey** – Konfigurace a data související se sčítači dopravy (ASD)
- **Portal** – Konfigurace portálů

- **Meteo** – Publikace meteorologických dat
- **TrafficData** – Publikace dopravních dat

Možných hodnot je více, jejich dokumentace je však mimo zaměření tohoto dokumentu a k naplnění požadované funkcionality nejsou potřeba.

4.2.10 Registrace nových entit

Rozhraní SDGW poskytuje dva způsoby registrace entity (zařízení):

- Předregistrace jako součást požadavku na enrollment
- Registrace nových entit pro schválenou Enrollment Identity

4.2.10.1 Předregistrace jako součást požadavku na Enrollment

Entity je možné předregistrovat formou sady identifikátorů (**EnrollmentEntities**) typu **Device** definovaných jako součást **CSR**.

Schválením požadavku na Enrollment jsou systémem zadané identifikátory entit schváleny a přiřazeny ke schválené **Enrollment Identity**. Výsledkem procesu je, že pro **EI** a její entity lze vyžádat **auth** token.

4.2.10.2 Registrace nových entit pro schválenou Enrollment Identity

Nové entity lze pro schválenou **Enrollment Identity** registrovat voláním metody **RegisterEntitiesRsa()** nebo **RegisterEntitiesEc()**. Záleží na použité metodě šifrování v předchozích metodách při žádosti o Enrollment. Pro úspěšné volání metody je nutné vyžádat **auth** token, ale v tomto případě není validováno, zda ID entity je schválenou entitou nebo nikoliv – registrace entity lze volat i přestože identifikátor entity není explicitně uveden v předaném tokenu. Jde o jedinou metodu s takovým typem výjimky.

4.2.11 Autentizace

Pro autentizaci požadavků na API je potřeba do HTTP hlaviček přidávat **Access** token. Viz kapitola 5.2.3 Získání **auth** tokenu výše v dokumentu.

4.2.12 Pravidla pro nakládání s auth tokenem

Systém, který komunikuje s rozhraním SDGW musí při žádosti o, nebo práci s **auth** tokenem dodržovat následující pravidla:

- Žádá jen o identifikátory entit, které jsou aktivní a schválené pro komunikaci s rozhraním SDGW
- Žádá o vydání nového **auth** tokenu nejdříve po uplynutí 2/3 intervalu mezi vydáním aktuálního tokenu a jeho expirací
- Žádá o vydání **auth** tokenu s maximálním možným seznamem autorizovaných entit (dávkově) v každé žádosti – autorizace samostatného tokenu pro každou entitu není povoleno

- Používá **auth** token jen pro získání a aktualizaci dat, která odpovídají entitám vydaným v použitém **auth** tokenu

EI nesmí žádat o tokeny způsobem, ze kterého by vyplývalo, že:

- Nevyužívá maximální možnou dobu použitelnosti již vydaných tokenů
- Při žádosti nespecifikuje maximální možný výčet identifikátorů entit v každém tokenu – max 50 identifikátorů na jeden **auth** token
- Mění často výčet identifikátorů entit, který pro jednotlivé tokeny využívá, a tím je nucen generovat tokeny znovu a znovu
- Nevyužívá obnovení **Access** tokenu pomocí **Refresh** tokenu v době jeho platnosti

Porušování výše uvedených pravidel je vyhodnoceno jako defekt integrovaného systému, případně bezpečnostní incident, a může vést k dočasnému či trvalému zrušení autorizace **EI** nebo přístupu k rozhraní SDGW.

5 Provozní a technické požadavky na integrované systémy

Je povinností integrátora systémů, které komunikují s rozhraním SDGW, dodržovat následující technické požadavky a pravidla. **Jejich porušení je vyhodnoceno jako porušení smluvních povinností integrátora.** Tyto povinnosti musí být přeneseny také na případné subdodavatele a spolupracující subjekty.

5.1 Požadavky na integrovaný systém

Všechny součásti integrovaného systému (např. zařízení, služby, agenti) musí splnit následující:

- **Integrovaný systém má správný systémový čas na všech úrovních** – systémový čas je synchronizován s důvěryhodným a přesným zdrojem času (např. GPS, NTP server apod.) nebo jiným zdrojem velmi přesného času, a je tak zajištěna správnost systémového času v každém okamžiku
- **Integrovaný systém je odolný vůči úniku citlivých údajů a dat** – generované privátní klíče, získané certifikáty, **auth** tokeny, klíče, hesla a další citlivé parametry (**secrets**) neopustí prostředí systému autorizovaného pro komunikaci s rozhraním SDGW. Tento typ dat je ukládán výhradně do šifrovaných a zabezpečených úložišť. Citlivé údaje nejsou zapisovány do logů.
- **Přenos dat mezi jednotlivými částmi integrovaného systému je zabezpečen** – všechny části integrovaného systému jsou při interní (vzájemné) i externí komunikaci zabezpečeny proti odposlechům a podvržení dat. Pro komunikaci se zařízeními a okolními službami je využíváno výhradně zabezpečení na úrovni transportní vrstvy (**TLS/SSL**), nebo zabezpečení pomocí **payload security**. Taková komunikace je vždy implementována v souladu s obecně závaznými a doporučenými postupy známými v okamžiku oživení integrovaného systému. Integrované systémy nesmějí být integrovány s rozhraním SDGW, pokud některé jejich části obsahují známé nebo potenciálně využitelné bezpečnostní zranitelnosti.
- **Hesla nejsou využívána pro ověřování uživatelů, a pokud už mají rozumnou sílu** – části integrovaného systému, které podporují bezpečnější metody ověření, než jsou hesla, nesmí pro ověření uživatelů hesla používat. V případech, kde toto není možné technicky zajistit, musí taková hesla splňovat následující parametry. Heslo nebude možné žádným z uživatelů změnit na variantu, která těmto požadavkům neodpovídá.
 - Minimálně 12 znaků
 - Jedno velké písmeno
 - Jedno malé písmeno
 - Jedna číslice
 - Jeden speciální znak

5.2 Autentizace, autorizace a důvěra

Provozovatelé a dodavatelé všech systémů, aplikací a služeb, které se integrují s rozhraním SDGW, zajistí, aby data z dodaných částí nebylo možné získat nebo modifikovat bez autentizace ani autorizace. Zařízení odmítnou změny v případech, kdy uživatel není autentizován nebo pro takovou změnu nemá uživatel oprávnění.

Výjimkou jsou pouze přístupy k datům a data, která ze své povahy musí být dostupná bez autentizace – např. data poskytovaná pomocí protokolu SNMP.

5.3 HSM

Služba **Enrollment Identity**, která je součástí integrovaného systému, musí mít klíče vygenerovány a bezpečně uloženy v **HSM** (Hardware Security Module).

Kritické části integrovaného systému, obsahující citlivá data, musí být zajištěny pomocí **Tamperu**, který zabrání úniku klíčů, pinů či hesel pro autorizaci **HSM**, autentizačních tokenů, a dalších citlivých informací z aplikací, služeb a zařízení, a z prostředí služby **Enrollment Identity**.

Zadavatel může v jednotlivých případech udělit výjimku a **HSM** pro konkrétní integraci nepožadovat. Integrátor však musí žádost odůvodnit a formou závazné bezpečnostní dokumentace deklarovat, jaká jiná opatření efektivně zabraňují úniku klíčů, jaká jsou rizika úniku konkrétních typů informací, jaká opatření budou provedena v případě úniku nebo kompromitace takových informací, a jak je jejich případný únik nebo zneužití monitorováno.

Ztráta či narušení **HSM**, porušení **Tamper** ochrany a další případné útoky. je integrátor povinen neprodleně hlásit provozovateli rozhraní SDGW a centrálního systému formou bezpečnostního incidentu.

Nad rámec této povinnosti, všechny aplikace, zařízení a jiné části integrovaného systému, hlídají porušení komponenty **Tamper** a její stav reportují se severitou **Critical** do rozhraní SDGW.

5.4 Auditní logy

Přístup i pokusy o přístup k jednotlivým částem integrovaného systému nebo datům, stejně jako uživatelské nebo systémové akce, je nutné auditovat. Auditní log musí být provozován minimálně v souladu s požadavky definovanými NÚKIB.

5.5 Bezpečnostní incidenty

Integrátor je provozovateli rozhraní SDGW a centrálního systému povinen, bez zbytečného odkladu, hlásit všechna

- Narušení jednotlivých částí integrovaného systému – krádež, porušení obalu, porušení **Tamper** apod.
- Únik privátních klíčů (**Encryption**, **Verification**)
- Únik certifikátů nebo **auth** tokenů vystavených rozhraním SDGW

- Anomálie v rámci běhu jednotlivých částí integrovaného systému, které by, byť i nevýznamně, ohrožovaly provoz nebo důvěryhodnost dat jednotlivých částí integrovaného systému
- Průnik do sítí integrátora, jeho subdodavatelů či zpracovatelů dat, nebo s nimi spolupracujících subjektů; i přestože s tím na první pohled účel dodávky nebo integrace s rozhraním SDGW nebo centrálním systémem nesouvisí

5.6 Servisní zásahy

Integrátor je provozovateli rozhraní SDGW a centrálního systému povinen, bez zbytečného odkladu, oznámit servisní zásahy na integrovaném systému nebo jeho částech:

- Servisní zásahy mající vliv na komunikaci nebo integraci s rozhraním SDGW
- Aktualizace systémů a zařízení

5.7 Řešení rozporů mezi specifikací rozhraní SDGW a tímto dokumentem

V případě rozporu mezi informacemi uvedenými v tomto dokumentu a strojově čitelnou specifikací rozhraní SDGW (**.proto**), má vždy přednost strojově čitelná specifikace rozhraní SDGW (**.proto**) předávaná společně s tímto dokumentem. Jde zejména o rozpory v názvech metod, režimech komunikace, změnách datových modelů apod.

6 Integrace s modulem pro řízení přístupu (ACCM)

Modul ACCM je součástí centrálního systému C-ITS a slouží k řízení infrastruktury v oblastech s omezeným přístupem. Umožňuje ovládat a monitorovat zařízení, jako jsou například výsuvné sloupky, semaforey, detekční kamery, smyčky nebo senzory obsazenosti.

Tato kapitola vás provede integrací zařízení a komponent do modulu ACCM pomocí dostupných metod rozhraní SDGW.

6.1 Registrace infrastrukturních zařízení

Předtím než začnete komunikovat s rozhraním SDGW v kontextu modulu ACCM, musíte zaregistrovat zařízení, které bude integrováno do tohoto systému. Rozlišujeme dvě hlavní kategorie zařízení:

- **přístupová zařízení** (např. výsuvné sloupky, závory, semaforey), která mohou být ovládána ze systému,
- **detekční zařízení** (např. kamery, indukční smyčky, senzory), která do systému odesílají informace.

6.1.1 Registrace přístupových zařízení

Pro registraci přístupového zařízení postupujte následujícím způsobem:

1. Odešlete do rozhraní SDGW požadavek pomocí metody `PublishDeviceSpecification`.
2. V parametru `device_specification.type` nastavte hodnotu `ACCESS_BARRIER`.
3. Vyplňte podobjekt `device_specification.subtype.access_barrier`, kde uvedete typ zábrany.
4. Pokud má zařízení podporovat vzdálené ovládání, nastavte příznaky v poli `accm_capabilities.supported_operations`:

Pro více informací o ovládání zařízení systémem ACCM přejděte na kapitolu [Ovládání zařízení pomocí povelů](#).

- `supports_control` – umožňuje přijímat základní povel jako OTEVŘÍT/ZAVŘÍT (např. `OPEN`, `CLOSE`),
- `supports_block` – umožňuje zařízení zablokovat a zabránit vykonávání ovládacích povelů až do jeho odblokování (`BLOCK`, `UNBLOCK`),
- `supports_block_external_control` – umožňuje blokovat ovládání externími zdroji (např. tlačítka, GSM brány); ovládání pomocí povelů zasílaných z modulu ACCM je stále umožněno.

Příklad zprávy `PublishDeviceSpecification` (pro přístupovou zábranu s podporou ovládání).

```
PublishDeviceSpecificationRequest {
  device_identity {
    enrollment_identity_id: "ba3de717-e77c-4b27-a832-98caad917eab"
    device_id: "f6cbe6f9-c851-4c93-aa9e-a109cbbdc5fb"
  }
  device_specification {
    preferred_name: "Výsuvný sloupek 01"
```

```

description: "Přístupová zábrana na vjezdu z ulice A"
type: ACCESS_BARRIER
sub_type {
  access_barrier: BOLLARD
}
}
accm_specification_container {
  accm_capabilities {
    supported_operations {
      supports_control: true
      supports_block: true
      supports_block_external_control: true
    }
  }
}
}
}

```

6.1.2 Registrace detekčních zařízení

Pro registraci detekčního zařízení (např. smyčka nebo kamera) postupujte obdobně:

1. Odešlete požadavek pomocí metody `PublishDeviceSpecification`.
2. V parametru `device_specification.type` nastavte hodnotu `SENSOR`.
3. Vyplňte podobjekt `device_specification.subtype.sensor`, kde definujete typ senzoru a jeho vlastnosti.

Příklad zprávy PublishDeviceSpecification (pro senzor obsazenosti).

```

PublishDeviceSpecificationRequest {
  device_identity {
    enrollment_identity_id: "d598c159-5e6d-4d6e-9f3c-02b5a5ad0c00"
    device_id: "39c35167-c65e-446e-a95f-6ea0c108c234"
  }
  device_specification {
    preferred_name: "Senzor Vjezd 01"
    description: "Magnetometrický senzor na vjezdu do areálu"
    type: SENSOR
    sub_type {
      sensor: MAGNETOMETER
    }
  }
}

```

Po registraci zařízení může komunikovat se systémem ACCM podle určené funkce (reportování stavů, detekcí, příjem povelů apod.)

6.2 Zasílání provozních stavů

Pravidelně informujte systém o aktuálním stavu zařízení a jeho komponent pomocí metody `PublishDeviceStatus`.

6.2.1 Zasílání provozního stavu zařízení

Pro zaslání provozního stavu zařízení do systému ACCM postupujte následujícím způsobem:

1. Odešlete zprávu pomocí metody `PublishDeviceStatus`.
2. V těle zprávy vyplňte objekt `device_status`, který musí obsahovat pole `reported_status` s hodnotou vyjadřující aktuální provozní stav zařízení, např. `OPERATIONAL`.
3. Doporučujeme také vyplnit `device_location`, pokud je poloha zařízení známa.

Příklad zprávy PublishDeviceStatus (provozuschopný stav včetně polohy).

```
PublishDeviceStatusRequest {
  request_id: "b1e59c6f-8932-4fd3-a7f2-53e9ac9c0001"
  request_timestamp: 1720600000123
  device_identity {
    enrollment_identity_id: "78c81234-3da3-47e2-9462-5a1eb86d0000"
    device_id: "e2a22e27-9a16-473a-93ef-866d50d0013c"
  }
  device_status {
    status_timestamp: 1720600000123
    reported_status: OK
  }
  device_location {
    location_info {
      status: FIXED
      timestamp: 1720600000123
      position {
        latitude: 50.087451
        longitude: 14.420671
        altitude {
          value: 192.3
        }
      }
    }
  }
}
```

6.2.2 Zasílání stavu přístupové zábrany

Zasílejte do systému aktuální stav přístupového zařízení (např. výsuvného sloupku nebo závory) pomocí metody `PublishDeviceStatus`.

1. Odešlete zprávu pomocí metody `PublishDeviceStatus`.
2. Vyplňte pole `access_barrier_status_container`, kde uvedete aktuální stav.
3. Vyplňte pole `device_location`, pokud je dostupná poloha zařízení.

Příklad zprávy PublishDeviceStatus (stav přístupové zábrany včetně stavu a polohy).

```
PublishDeviceStatusRequest {
  request_id: "b8077934-13ec-48df-a7be-b71e9d7b24ab"
  request_timestamp: 1720593612345
  device_identity {
    enrollment_identity_id: "b85af1b7-0150-4cbe-95b2-5f083c3010b4"
    device_id: "3e97bb15-2e6a-4c71-bb39-b0c5c626b3a3"
  }
}
```

```

}
device_status {
  status_timestamp: 1720593612345
  reported_status: OK
}
access_barrier_status_container {
  status: OPEN
}
device_location {
  location_info {
    status: FIXED
    timestamp: 1720593612345
    position {
      latitude: 50.088042
      longitude: 14.403964
    }
  }
}
}
}

```

6.3 Zasílání detekcí objektů

Reportujte do systému ACCM všechny události detekované zařízeními na lokalitě (např. detekce přítomnosti vozidla).

6.3.1 Zasílání základní detekce objektu

Každou novou detekci odešlete pomocí zprávy `PublishDetection`, která musí obsahovat následující údaje:

- `detection_id` – jedinečný identifikátor detekce vedený v integrovaném systému
- `detected_at` – čas vzniku detekce
- `device_identity` – identita zařízení, ze kterého detekce vznikla
- `location_info` – poloha detekce
- `object_detection_container` (*volitelné*) – typ detekovaného objektu (např. vozidlo)

Příklad zprávy `PublishDetection`.

```

PublishDetectionRequest {
  detection_id: "3f44d8e7-93f4-4bce-8c6a-8b1c146672f1"
  detected_at: 1720593612345
  device_identity {
    enrollment_identity_id: "78c81234-3da3-47e2-9462-5a1eb86d0000"
    device_id: "9b5c4b93-71b6-4891-a3f2-dfd2d73484fc"
  }
  location_info {
    coordinates {
      latitude: 50.087451
      longitude: 14.420671
      altitude {
        value: 230.5
      }
    }
  }
}

```

```
}
object_detection_container {
  object_category: VEHICLE
  object_type: PASSENGER_CAR
}
```

6.3.2 Doplnění identifikátoru k detekci

Pokud integrovaný systém, jako součást detekce objektu, zaznamenal také alespoň jeden identifikátor (např. rozpoznávání registrační značky vozidla), je po zaznamenání detekce možné doplnit jeden nebo více identifikátorů (RZ, identifikační karta apod.)

Identifikátor zaznamenejte k detekci pomocí metody `AddDetectionIdentifier`. Každý identifikátor přidávejte pomocí samostatné zprávy.

1. Odesílejte jeden nebo více požadavků `AddDetectionIdentifier` – pro každý identifikátor jeden požadavek.
2. Vyplňte pole `detection_id` a `detection_identifier`, které obsahuje čas detekce, typ identifikátoru a jeho konkrétní hodnotu.

Příklad zprávy `AddDetectionIdentifier`.

```
AddDetectionIdentifierRequest {
  detection_id: "3f44d8e7-93f4-4bce-8c6a-8b1c146672f1"
  detection_identifier {
    timestamp: 1720593612345
    identifier_type: LICENSE_PLATE
    identifier: "3AB1234"
  }
}
```

6.3.3 Připojení souboru k detekci

Pokud má integrovaný systém k detekci dostupné další informace (např. fotografie), je možné tyto informace nahrát na server a následně přiřadit k zaznamenané detekci.

Po nahrání souboru do systému ACCM jej připojte k detekci pomocí metody `AddDetectionFileAttachment`. Opakováním požadavku připojíte k jedné detekci více souborů.

Pro informace o nahrávání souboru přejděte na kapitolu [Nahrávání souborů \(např. fotografie vozidla\)](#).

1. Odesílejte jeden nebo více požadavků `AddDetectionFileAttachment` – pro každý připojovaný soubor jeden požadavek.
2. Uveďte ID detekce, ID nahraného souboru a volitelně čas připojení nebo metadata o médiu.

Nepřipojené (volné) soubory budou po několika hodinách automaticky odstraněny.

Příklad zprávy `AddDetectionFileAttachment`.

```
AddDetectionFileAttachmentRequest {
  detection_id: "3f44d8e7-93f4-4bce-8c6a-8b1c146672f1"
  timestamp: 1720593640000
  uploaded_file_id: "file-abc123"
```

```
media_container {
  content_type: LICENSE_PLATE
}
```

6.4 Ovládání zařízení pomocí povelů

Zařízení, jako jsou závory nebo výsuvné sloupky, lze ovládat přímo z modulu ACCM. Tato sekce popisuje, jak modulu ACCM definovat podporované povel, jak z modulu ACCM přijímat povel, a jak informovat o výsledku jejich vykonání.

6.4.1 Definice podporovaných povelů

Seznam podporovaných povelů definujete při registraci zařízení. Podrobnosti naleznete v kapitole [Registrace přístupových zařízení](#).

6.4.2 Příjem povelů ze systému ACCM

Pro příjem příkazů implementujte duplexní stream `SubscribeAccmCommands`, který je součástí služby `AccmCommandService`.

Při navázání spojení je nutné v první zprávě zaslat poslední zpracovanou verzi v parametru `device_subscriptions`, pro kterou chcete získávat příkazy. Pokud se jedná o první inicializaci zařízení (verze 0), server následně odešle všechny relevantní příkazy s verzí vyšší, než kterou jste uvedli. Tento proces odpovídá tzv. **Outbox Pattern**. Po úspěšné inicializaci začnete přijímat zprávy typu `SubscribeAccmCommandsItemResponse`, které obsahují:

- `request_id` – identifikátor požadavku
- `device_identity` – identita zařízení (enrollment a device ID)
- `timestamp` – čas přijetí zprávy
- `event_info` – informace o události (typ a verze)
- `command` – příkaz typu `AccmControlCommandPayload`

Příklad přijaté zprávy s příkazem.

```
SubscribeAccmCommandsItemResponse {
  request_id: "b4d4a429-8379-4530-8ae3-d13907eb8f4e"
  device_identity {
    enrollment_identity_id: "c49b2978-8f14-4f7d-a1e7-8a878859c4e1"
    device_id: "a3769dc6-6b6c-4f8c-b11f-2f16c114219d"
  }
  timestamp {
    seconds: 1719990000
  }
  event_info {
    event_version_info {
      event_version_info: 1
    }
  }
  event_type: CREATED
```

```

}
command {
  command_id: "631fb7e0-2c12-47c0-b02e-dc9d8ec23457"
  command_type: OPEN
}
}

```

6.4.3 Zasílání výsledků zpracovaných povelů

Po vykonání příkazu odešlete výsledek do systému pomocí zprávy `SubscribeAccmCommandsItemRequest`.

Výsledek musí být uveden v poli `device_command_updates.updates.command_status`, které obsahuje:

Do objektu `confirmed_version_info` uveďte verzi zpracovaného příkazu. Tím serveru sdělíte, že tuto verzi může bezpečně zahodit.

- `command_id` – identifikátor původního příkazu
- `timestamp` – čas zpracování
- `command_status` – výsledek: `ACCEPTED`, `COMPLETED`, `FAILED`, `REJECTED` apod.
- `command_status_message_data` – (volitelně) doplňující zpráva nebo popis chyby

Příklad zprávy s výsledkem zpracování.

```

SubscribeAccmCommandsItemRequest {
  request_id: "c214d5e3-1df5-4322-931b-8cbfe6cfc331"
  timestamp: 1720600010000
  device_command_updates {
    updates {
      device_identity {
        enrollment_identity_id: "be6ec213-3f3e-4b4f-89cb-f30e236f6f73"
        device_id: "cd75e49a-0215-4a4f-b93a-0990d24a7fa4"
      }
      confirmed_version_info {
        event_version_info: 1
      }
      command_status {
        command_id: "fa63ce4b-2a58-4a59-8e5d-b3a55f35c9c6"
        timestamp: 1720600010000
        command_status: COMPLETED
        command_status_message_data {
          timestamp: 1720600010000
          message: "Barrier raised successfully"
          severity: OK
        }
      }
    }
  }
}

```

6.5 Zasílání informací o ovládání

Pokud je na zařízení zaznamenán povel k ovládání, pak, neohledě na to, odkud povel vznikl (ACCM, externí zdroj), informujte o takovém požadavku modul ACCM pomocí metody `PublishAccmControlEventLogItem`.

Použijte zprávu `PublishAccmControlEventLogItemRequest`, která musí obsahovat následující údaje v poli `event_log_item`:

- `timestamp` – čas události
- `command_type` – typ vykonaného příkazu (např. `RAISE_BARRIER`)
- `outcome` – výsledek provedení (`SUCCESS`, `FAILURE` apod.)
- `event_id` – jedinečný identifikátor události

Příklad zprávy `PublishAccmControlEventLogItemRequest`.

```
PublishAccmControlEventLogItemRequest {
  device_identity {
    device_id: "d5b8f527-f14e-4e1b-8c5d-84b27d92d3d9"
  }
  event_log_item {
    timestamp {
      seconds: 1720000000
    }
    command_type: RAISE_BARRIER
    outcome: SUCCESS
    event_id: "f7c3b872-31ff-44f3-886a-b5ab9c401c2e"
    correlation_id: "b01f56b4-7f98-44ea-9e7a-0b14a18051fb"
    affected_resource: "device/d5b8f527-f14e-4e1b-8c5d-84b27d92d3d9/control/manual"
    operation_authorization {
      authorization_identifier_type: USER_ID
      authorization_identifier: "6be22dd6-d574-4b7d-8100-12c54b962c94"
    }
    triggered_by {
      type: SERVER_COMMAND
      trigger_identifier: "b01f56b4-7f98-44ea-9e7a-0b14a18051fb"
    }
  }
}
```

Zprávu `PublishAccmControlEventLogItem` používejte pro záznam všech ovládacích událostí — jak těch, které byly iniciovány externím systémem (např. lokální jednotka, operátor), tak těch, které byly iniciovány systémem ACCM.

V případě, že událost byla spuštěna systémem ACCM, je nutné:

- do pole `triggered_by.type` vyplnit hodnotu `SERVER_COMMAND`
- do pole `triggered_by.trigger_identifier` uvést `command_id` odpovídajícího příkazu ze systému ACCM

6.6 Získávání konfigurace detekčních zón

Zařízení mohou být ovládána na základě detekcí v určitých zónách – například pokud vozidlo vjede do konkrétního prostoru. Pro zajištění správné funkce systému sledujte, zda se nezměnila konfigurace těchto detekčních zón, a vždy pracujte s jejich aktuální verzí.

1. Monitorujte změny verzí kolekcí s typem `data_collection_type.DETECTION_ZONE` pomocí metody `SubscribeSystemDataCollectionVersioning`.

2. Po detekci změny verze načtete aktuální konfiguraci pomocí metody `GetDetectionZones`.

Systém musí verze kolekcí sledovat průběžně, aby vždy pracoval s aktuální konfigurací detekčních zón.

Příklad požadavku `GetDetectionZonesRequest`.

```
GetDetectionZonesRequest {
  device_identity {
    enrollment_identity_id: "9f3e92e1-ec4f-4f4a-89c2-5e4d4e44b000"
    device_id: "3d7bce2e-2ba0-4fd4-88cf-c3fe7b961f33"
  }
}
```

6.7 Získávání konfigurace časových plánů

Zařízení mohou fungovat dle časových plánů definovaných v modulu ACCM. Tyto plány určují, kdy má být zařízení v otevřeném, zavřeném nebo jiném definovaném stavu.

Pro zajištění aktuálnosti konfigurace:

1. Monitorujte změny verzí kolekcí typu `data_collection_type.ACCM_CONTROL_SCHEDULE` pomocí streamu `SubscribeSystemDataCollectionVersioning`.
2. Po zjištění změny verze načtete aktuální seznam plánů pomocí metody `GetAccmControlSchedules`.

Načtené plány aplikujte pouze v případě, že je zařízení odpojeno od systému ACCM. Pokud je připojeno, je řízeno přímo serverem.

Příklad požadavku `GetAccmControlSchedulesRequest`.

```
GetAccmControlSchedulesRequest {
  device_identity {
    enrollment_identity_id: "c61baf5c-ef6e-42a5-b2d5-1f9cb4e6d411"
    device_id: "barrier-42"
  }
}
```

6.8 Zasílání událostí ze zařízení

Zařízení může během provozu generovat různé události – například chyby, restart, překážku v dráze nebo servisní zásah. Tyto události odesíláte do systému ACCM pomocí metody `PublishDeviceEventLogItem`.

1. Odešlete požadavek `PublishDeviceEventLogItemRequest`, ve kterém uveďte identifikaci zařízení (`device_identity`), případně komponenty (`component_id`), a detailní popis události v poli `event_log_item`.
2. Každá událost musí obsahovat čas, typ, závažnost a případně další informace.

Příklad zprávy `PublishDeviceEventLogItemRequest`.

```
PublishDeviceEventLogItemRequest {
  device_identity {
    enrollment_identity_id: "7ae7d76e-5f59-463e-a5b3-3bc52adf2f2d"
  }
}
```

```

    device_id: "a24d87bb-fbd3-4f75-880b-646bb14b94f1"
  }
  component_id: "c5f6c94a-4f1d-4b7a-909f-9d9c93b55e68"
  event_log_item {
    timestamp {
      seconds: 1720001000
    }
    type: "DEVICE_RESTART"
    severity: WARNING
    message: "Device restarted unexpectedly"
    event_id: "df5394c1-18dc-49ad-97b2-bfdd7982c9d1"
  }
}

```

6.9 Nahrávání souborů (např. fotografie vozidla)

Zařízení může generovat soubory (např. fotografie nebo video), které je potřeba nahrát do rozhraní SDGW, a následně je přiřadit ke konkrétní detekci nebo události.

Pro nahrání souboru použijte duplexní stream `UploadFile` ze služby `UploadService` a postupujte následovně:

1. Odešlete první zprávu typu `UploadFileItemRequest` s vyplněným polem `file_metadata`. Uveďte identitu zařízení (`device_identity`), GUID souboru (`file_id`), MIME typ, velikost a případně další metadata.
2. Počkejte na odpověď serveru. Pokud server zprávu přijme, odpověď bude obsahovat objekt `transfer_definition` s pravidly pro přenos (např. maximální velikost bloku a časový limit pro opakování).
3. Začněte odesílat obsah souboru po blocích. Každý blok zabalte do nové zprávy `UploadFileItemRequest` s vyplněným polem `data`, které obsahuje bajtový offset a binární data. Bloky odesílejte postupně – vždy odešlete jeden blok a vyčkejte na potvrzení (`confirmed_position`).
4. Pokud server potvrdí přijetí bloku, pokračujte odesláním dalšího bloku.
5. Pokud server blok nepotvrdí do času uvedeného v poli `transfer_definition.wait_before_retry_seconds`, odešlete tento blok znovu.
6. Nakonec odešlete poslední zprávu s prázdným obsahem (`data: ""`) a správným `byte_offset`, čímž označíte konec přenosu.

Pokud soubor nepřipojíte k žádné detekci nebo jiné entitě (např. pomocí `AddDetectionFileAttachment`), bude po několika hodinách automaticky odstraněn.

Příklad zpráv pro nahrávání souboru.

```

// Počáteční zpráva s metadaty
UploadFileItemRequest {
  file_metadata {
    source_identity {
      enrollment_identity_id: "abc123"
      device_id: "camera-01"
    }
  }
  file_id: "file-abc123"
  mime_type: "image/jpeg"
}

```

```
    file_name: "snapshot.jpg"
    file_extension: "jpg"
    file_size: 123456
  }
}

// Blok binárních dat
UploadFileItemRequest {
  data {
    byte_offset: 0
    data: "<binární obsah>"
  }
}

// Poslední (prázdná) zpráva pro ukončení přenosu
UploadFileItemRequest {
  data {
    byte_offset: 123456
    data: ""
  }
}
```