

Отдел за безопасност на движението

3M, сграда 235-3A-09

Ст. Пол, Минесота 55133-3225 [www.3M.oom/mvss](http://www.3M.oom/mvss)

### 3M™ КАМЕРА СЪС СРЕДНА СКОРОСТ

### WATCH LIST PLUGIN ISSUE 2.1

### СПИСЪК С ИЗВЪРШЕНИТЕ ПРОМЕНИ

|  |  |  |
| --- | --- | --- |
| ИЗДАНИЕ | ДАТА | КРАТКИ ДАННИ ЗА ИЗВЪРШЕНИТЕ ПРОМЕНИ |
| Чернова | 31/05/2012 | Първоначална версия |
| 2.0 | 16/08/2012 | Допълнителни изисквания, добавени от клиента |
| 2.1 | 15/01/2014 | Актуализиране на 3М номенклатурата за продукти |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Настоящият документ представлява последното издание, показано в таблицата по-горе.

Автори: Артур Бидиан

**СЪДЪРЖАНИЕ**

1. [Въведение 4](#_TOC_250008)
2. [Дефиниции 5](#_TOC_250007)
3. [Вход (Списък за наблюдение) 6](#_TOC_250006)
4. [Изход при засичане на попадение 8](#_TOC_250005)
   1. [Таблицата PlateFeed 9](#_TOC_250004)
   2. [Засичане на попадение: Таблиците PlateWatchListDetection и PlateWatchListDetectionImages 9](#_TOC_250003)
5. [Приложение A – Примерен вход 12](#_TOC_250002)
6. [Приложение B – Схеми на базите данни 13](#_TOC_250001)
7. [Приложение C – Цифров подпис на запис в PlateWatchListDetection 15](#_TOC_250000)

# Въведение

Настоящият документ дефинира интерфейса до/от сървърния софтуер на 3M™ камерата със средна скорост за:

* + даване на възможност за експортиране на опростен, лек стрийминг на информация за засичане на регистрационни номер (което изключва изображения) на 3-ти страни;
  + позволява на 3-ти страни да предоставят списъци за проследяване на регистрационни номера, а на 3M™ камерата със средна скорост да експортира информация за засичане на нарушения, включително и изображения

3-ти страни имат достъп единствено до посоченият и описан в настоящия документ интерфейс (по-конкретно, дефинирания вход и изход в този документ – вижте съответните раздели), а употребата на всякакви други данни и структури на вътрешната станция (3M™ камерата със средна скорост) от 3-ти лица не се поддържа и може да доведе до получаването на неочаквани резултати. Такива действия биха представлявали собствен риск за 3-тите страни и 3М не носи отговорност за употреба, която не поддържа. Естеството на такъв вид разширяване на 3M™ камерата със средна скорост би поставило софтуера извън границите на типовото одобрение, така че 3-ти страни трябва да го имат предвид.

Настоящият документ не описва функциите на 3M™ камерата със средна скорост, а се предполага, че те вече са познати на потребителя. Накратко, 3M™ камерата със средна скорост предоставя средна скорост за дистанционно изпълнение в мрежа от камери.

Олекотеният експорт на уловени регистрационни номера и списъкът за наблюдение са допълнителни функционалности към изпълнението на средна скорост, което се предоставя от софтуерния пакет на 3M™ камерата със средна скорост.

# Дефиниции

* + ERCU сървър – средство за възстановяване и контрол на доказателства, сървърен софтуер – един от двата основни компонента на сървърния софтуер на 3M™ камерата със средна скорост
  + SM сървър – система за управление на сесиите, сървърен софтуер - един от двата основни компонента на сървърния софтуер на 3M™ камерата със средна скорост
  + Вътрешна станция – основният сървърен софтуер на 3M™ камерата със средна скорост, който се състои от своето ядро и от сървърите ERCU и SM
  + VRN – регистрационен номер на превозно средство, познато и като регистрационна табела
  + Списък за наблюдение – списък с превозните средства (VRN), които представляват интерес
  + Попадение – събитие, при което превозно средство от списъка за наблюдение е уловено от камерата и е представено на вътрешната станция
  + Доказателствен запис – състои се от данните, получени от камерата, включващи VRN, времеви печати на заснемането (GPS и календарно време), заснети изображения.

# Вход (Списък за наблюдение)

Вътрешната станция очаква входът да бъде във формата на два текстови файла, с предварително уточнени имена. Двата файла, които сървърът (вътрешната станция) очаква да открие, са предназначени да предоставят входа за списъка с регистрационни номера, които ще се наблюдават (напр. *„списък за наблюдение – регистрационни номера“*), както и входа за избор на конкретни камери (позната и като обектна идентификация на камерата), които да бъдат наблюдавани (напр. *„списък за наблюдение - камери“*) за списъка за наблюдение на регистрационни номера. Очаква се тези файлове да бъдат поставени на добре познато място (в папка *“conf“* на ERCU). Тези файлове осигуряват динамичен вход за списъка за наблюдение на регистрационните номера на вътрешната станция. При засичане на промени в тези файлове, вътрешната станция ги зарежда в паметта. Списъците за наблюдение за регистрационните табели и за камерите са разделени умишлено на 2 отделни файла, за да се позволи лесната динамична промяна на всеки от тях, без да се засяга другия: например, файлът с камерите за списъка за наблюдение може да бъде малък и извършване на промяна в него означава, че няма да има нужда да се генерира отново (и да се зареди отново) и целия списък с регистрационни табели (който би могъл да бъде голям).

Файлът за списък за наблюдение на камерите може да бъде пропуснат. В този случай вътрешната станция ще следи списъка за наблюдение на регистрационните табели, независимо от това, от коя камера са изпратени номерата. Ако е предоставена конкретна камера за списъка за наблюдение, сървърът ще наблюдава само тези камери/обектни идентификации, които са предоставени във файла: само за тези камери сървърът ще предостави резултат при засечено попадение. Това, обаче, няма отражение върху редовните данни за регистрационни табели, които сървърът експортира: данните ще бъдат изпратени от всички камери, независимо от наличието на файл с камери за списъка за наблюдение.

От файла с камерите за списъка за наблюдение се очаква да следва следните правила:

* + Очаква се да започва с ред, в който се съдържа само изразът “*[camera watchlist]”* (без кавичките), който да завършва с нов ред (‘\n’): това гарантира второ ниво на валидиране на съдържанието на файла. Всеки ред след този ще бъде считан за обектна идентификация на камера и ще бъде зареждан;
  + Очаква се на всеки ред да има по една обектна идентификация на камера (получена от базата данни на ERCU, в частност, тя трябва да отговаря на полето “*обектна идентификация*” в таблицата *Камера*). Счита се, че един ред приключва след въвеждането на ('\n');
  + Честотата на промяна на файла не може да надвишава една промяна на минута. По-чести промени не се поддържат и могат да доведат до неочаквано поведение на софтуера;
  + Файлът трябва винаги да представя пълния набор от камери за списъка за наблюдение. Актуализиране само на промени не се поддържа. След извършване на актуализация се счита, че файлът съдържа пълния набор от камери за списъка за наблюдение и ще бъде използван в този вид за напред;
  + Вътрешната станция съхранява списъка за наблюдение в паметта и не съхранява списъка в никоя от таблиците на базата си данни;

От файла с регистрационни номера за наблюдение се очаква да следва следните правила:

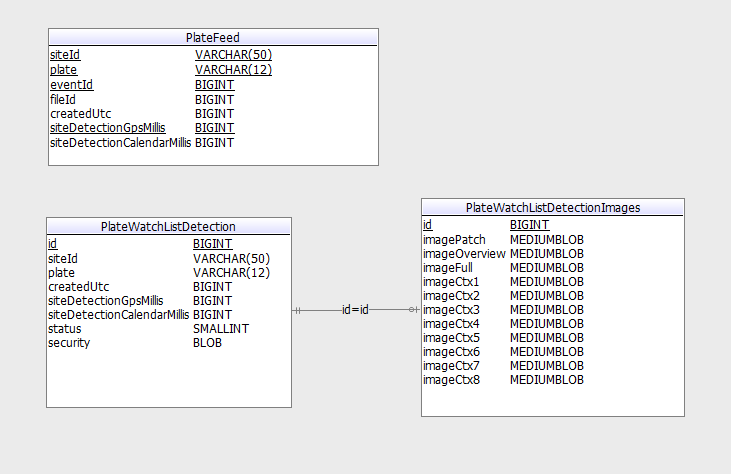
* + Очаква се да започва с ред, в който се съдържа само изразът “*[plate watchlist]”* (без кавичките), който да завършва с нов ред (‘\n’): това гарантира второ ниво на валидиране на съдържанието на файла. Всеки ред след този ще бъде считан за регистрационен номер на превозно средство и ще бъде зареждан. Счита се, че редът завършва с ('\n');
  + Вътрешната станция не може да поддържа повече от 5 млн. въведени регистрационни номера в този файл, които ще бъдат наблюдавани за извършване на промени. Този файл представлява базата данни на списъка за наблюдение, която сървърът/софтуерът на вътрешната станция ще наблюдава и ще зарежда отново, при засичане на промени;
  + Честотата на промяна на файла не може да надвишава една промяна на минута. По-чести промени не се поддържат и могат да доведат до неочаквано поведение на софтуера;
  + Файлът трябва винаги да представя пълния списък за наблюдение. Актуализиране само на промени не се поддържа. След извършване на актуализация се счита, че файлът съдържа пълния списък за наблюдение и ще бъде използван в този вид за напред;
  + Вътрешната станция ще съхранява списъка за наблюдение в паметта, а не запазва списъка в никоя от таблиците. Не се поддържа взаимодействие с различни от поддържаните таблици от базата данни;

# Изход при засичане на попадение

Изходът във вътрешната станция при намерени отговарящи номера от списъка за наблюдение и препращането на олекотени данни за регистрационните табели (в които се изключват изображенията) от вътрешната станция се извършва с помощта на предварително съгласувани таблици от базата данни. Таблиците се намират в специално разработена и създадена база данни, която се различава от вътрешната база данни на вътрешната станция. Нарича се “*ercu\_plugin*” и работи на MySQL сървър за бази данни. Отговорност на 3-тата страна е да управлява и да се грижи за базата си от данни. Препоръчва се периодично архивиране на базата данни, за да се контролира размера на данните. Вътрешната станция трябва да бъде изключвана при архивиране на базата данни *ercu\_plugin*, ето защо е препоръчително това да се прави, когато няма много голям трафик (напр. късните часове на нощта или ранните часове на деня). Тези таблици са:

* + PlateFeed – в тази таблица се съхраняват регистрационните номера, които пристигат в сървъра. Тези данни са олекотени данни за регистрационните табели и не съдържат изображенията, уловени от камерата
  + PlateWatchListDetection и PlateWatchListDetectionImages – тези две таблици (заедно) съдържат резултата от попаденията на регистрационни номера

Следната фигура илюстрира таблиците (схемата на базата данни за таблици е поместена и в Приложение В):



## Таблицата PlateFeed

Таблицата *PlateFeed* съдържа регистрационните номера, които са уловени (олекотени данни без изображения) от вътрешната станция и са запазени в тази таблица.

Таблицата PlateFeed се състои от:

|  |  |
| --- | --- |
| siteId | Обектната идентификация на камерата, максимален брой от 50 символа. Това трябва да бъде една от обектните идентификации на камерата, конфигурирани в рамките на базата данни на ERCU. |
| Plate | Текстът на регистрационния номер, максимален брой от 12 символа. |
| eventId | Идентификация на вътрешно събитие на камерата за уловен регистрационен номер |
| fileId | Идентификация на вътрешен файл на камерата за оловен регистрационен номер |
| createdUtc | UTC времеви печат при създаване на запис за регистрационен номер от сървъра |
| siteDetectionGpsMillis | GPS времето в милисекунди при засичане на регистрационния номер от камерата |
| siteDetectionCalendarMillis | Вътрешното календарно UTC време в милисекунди при засичане на регистрационния номер от камерата |

Първичният ключ на тази таблица се състои от четворката (`siteId`, `plate`,`siteDetectionGpsMillis`, `eventId). Очаква се таблицата *PlateFeed* да бъде управлявана от потребител на 3-та страна. След като един запис е записан в таблицата, вътрешната станция не променя повече данните и отговорността за управление и архивиране на данните се носи изключително от 3-тата страна.

## Засичане на попадение: Таблиците PlateWatchListDetection и PlateWatchListDetectionImages

Таблица PlateWatchListDetection, заедно с таблица PlateWatchListDetectionImages съдържа попаденията, засечени от вътрешната станция. В тях се съдържат записи, които в момента се обработват от вътрешната станция.

Попадение в рамките на вътрешната станция може да бъде намерено в едно от следните състояния (да не се бърка със състоянията на входовете в таблицата PlateWatchListDetection):

* Засечено и подадено във вътрешната станция;
* Подадено в камерата; камерата обработва заявката и изпраща изхода (записа изпратен или неуспешно изпращане);
* Получено от вътрешната станция от камерата;

При някои сценарии заявките за доказателствения запис не могат да бъдат изтеглени. Това може да се дължи на няколко причини: доказателственият запис не може да бъде открит (например, ако камерата е трябвало да запише върху доказателствения запис, поради изчерпване на пространството за съхранение), изтичане на доказателствения запис и т.н.

Тези грешки (липса на доставка на доказателствен запис), обаче, не се докладват в таблицата PlateWatchListDetection в тази версия на продукта, за да не се въздейства върху нормалното функциониране и работата на вътрешната станция.

Ако дадено (или вход в тази таблица) попадение бъде управлявано от вътрешната станция, то може да бъде намерено в едно от следните състояния, които директно се отразяват от стойността на полето “*status*” в тази таблица:

* стойността на *status* е 1 – засечено е попадение на регистрационен номер и заявка за доказателствен запис за съвпадащия регистрационен номер (който съответства на списъка за наблюдение) изчаква в системата като цяло (това може да бъде вътрешната станция или заявката може да е стигнала до самата камера). Към този момент в таблицата PlateWatchListDetectionImages няма съответстващ вход (тъй като все още нямам изображенията).
* стойността на *status* е 5 – изтеглен е доказателственият запис и съответните полета в таблицата са попълнени, готови за допълнително обработване от 3-тата страна. След това записът е готов за обработка от 3-тата страна, а стойността на полето *status* може свободно да бъде променена, така че отразява собствени етапи на обработка на 3-тата страна.

Вътрешната станция запазва за себе си първите 5 числа, които да служат за собствен индикатор на статуса за засичане на списъка за наблюдение, от 0 до 5 включително, където 0 – непознат, 1 – засечено е попадение на регистрационен номер, 5 – доказателственият запис е изтеглен от камерата и данните са готови за обработка от 3-тата страна. Вътрешната станция не дава гаранция за междинни справки за статуса в тази таблица (напр. не дава гаранция, че ще покаже статуса на заявката при достигането й до камера, или изхода на заявката).

Докато се управлява от вътрешната станция по време на всички етапи даден вход в таблицата е защитен с цифров подпис. Всички съответстващи данни се изчисляват в този подпис, а публичният ключ за потвърждаване на целостта на данните от 3-ти лица се експортира в предварително съгласуван файл, за да се използва от 3-тата страна. Така се гарантира целостта на данните, които се експортират до 3-тата страна. За повече подробности относно изчисляването на подписа, моля, вижте Приложение С.

Управлението и поддържането (архивиране при нужда) на двете таблици, а именно *PlateWatchListDetection* и *PlateWatchListDetectionImages,* е отговорност изключително на 3-тата страна. С времето таблиците могат да се разширят и е отговорност на клиента периодично да архивира данните, за да поддържа размера на таблиците в размер, подходящ за нормална работа.

Таблицата PlateWatchListDetection се състои от:

|  |  |
| --- | --- |
| id | Уникален, автоматично увеличаващ се идентификационен номер за таблицата |
| siteId | Уникална обектна идентификация на камерата, която е засекла регистрационния номер |
| plate | Текстът на регистрационната табела, който трябва да съвпадне със списъка за наблюдение, максимален брой символи – 12 |
| createdUtc | Създаване & засичане на UTC време от вътрешната станция в милисекунди. Различава се от времето на улавяне на камерата. |
| siteDetectionGpsMillis | GPS времето в милисекунди при засичане на регистрационния номер от камерата |
| siteDetectionCalendarMillis | Вътрешното календарно UTC време в милисекунди при засичане на регистрационния номер от камерата |
| status | Статусът на обработваните записи в базата данни |
| security | Подписът на записа, изчислен и попълнен от вътрешната станция при промяна или създаване на записа |

Таблицата PlateWatchListDetectionImages се състои от:

|  |  |
| --- | --- |
| id | Уникалният идентификационен номер за таблицата, който по уникален начин се свързва със завписа в таблица *PlateWatchListDetection* |
| imagePatch | JPEG изображение на табелата от доказателствения запис, получен от камерата |
| imageOverview | JPEG изображение на общ план от доказателствения запис, получен от камерата |
| imageFull | Пълното JPEG изображение от доказателствения запис, получен от камерата |
| imageCtx1 | Едно от максимално възможните 8 контекстни изображения, които могат да бъдат получени от камерата. Заб.: в случая на 3M™ камерата със средна скорост, този брой е 1 (камерата предоставя 1 контекстно изображение) |
| imageCtx2 | Едно от максимално възможните 8 контекстни изображения, които могат да бъдат получени от камерата. Заб.: в случая на 3M™ камерата със средна скорост, това поле е нула |
| imageCtx3 | Същото като по-горе |
| imageCtx4 | Същото като по-горе |
| imageCtx5 | Същото като по-горе |
| imageCtx6 | Същото като по-горе |
| imageCtx7 | Същото като по-горе |
| imageCtx8 | Същото като по-горе |

За всеки запис в таблица PlateWatchListDetection може да има нула или един запис, свързан с таблица PlateWatchListDetectionImages. Липсата на записа за таблица PlateWatchListDetectionImages показва, че все още не са получени изображения, или че такива не могат да бъдат получени. Съществуващ запис, обаче, би предоставил набор от изображения, които съответстват на регистрационния номер.

# Приложение A – Примерен вход

Следва съдържанието на типичен файл с набор от камери за списък за наблюдение (“*watchlist-cameras*“):

[camera watchlist] 12283\_1

12284\_1

12285\_1

12286\_1

Редовете под [camera watchlist] представляват обектните идентификации на камерите, които могат да бъдат намерени в таблица Камера на вътрешната базата данни на ERCU (полето *siteid*). Тези камери ще бъдат наблюдавани за предоставения списък с регистрационни табели. Ако липсва такъв файл, всички камери в рамките на вътрешната станция ще бъдат наблюдавани за предоставения списък с регистрационни табели.

Следва съдържанието на типичен файл с регистрационни номера за списъка за наблюдение (“*watchlist-plates*”):

[plate watchlist] ABCD123

EFGH456 IJKL789

Редовете след входа *[plate watchlist]* представляват регистрационните номера (VRNs), които ще бъдат наблюдавани от вътрешната станция.

# Приложение B – Схеми на базите данни

SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED; SET GLOBAL TRANSACTION ISOLATION LEVEL READ COMMITTED;

/\* ===================================\*/

DROP TABLE IF EXISTS `ercu\_plugin`.`PlateWatchListDetectionImages`; DROP TABLE IF EXISTS `ercu\_plugin`.`PlateWatchListDetection`;

DROP TABLE IF EXISTS `ercu\_plugin`.`PlateFeed`;

CREATE TABLE `ercu\_plugin`.`PlateFeed` (

`siteId` VARCHAR(50) NOT NULL

, `plate` VARCHAR(12) NOT NULL

, `eventId` BIGINT NOT NULL

, `fileId` BIGINT

, `createdUtc` BIGINT NOT NULL

, `siteDetectionGpsMillis` BIGINT NOT NULL

, `siteDetectionCalendarMillis` BIGINT NOT NULL

, PRIMARY KEY (`siteId`, `plate`, `siteDetectionGpsMillis`, `eventId`)

);

CREATE TABLE `ercu\_plugin`.`PlateWatchListDetection` (

`id` BIGINT NOT NULL AUTO\_INCREMENT

, `siteId` VARCHAR(50) NOT NULL

, `plate` VARCHAR(12) NOT NULL

, `createdUtc` BIGINT NOT NULL

, `siteDetectionGpsMillis` BIGINT NOT NULL

, `siteDetectionCalendarMillis` BIGINT NOT NULL

, `status` SMALLINT

, `security` BLOB

, PRIMARY KEY (`id`)

)ENGINE=InnoDB;

ALTER TABLE `ercu\_plugin`.`PlateWatchListDetection` MODIFY COLUMN

`siteDetectionGpsMillis` BIGINT NOT NULL

COMMENT 'The timestamp of the detection by the camera (camera''s GPS derived time)';

ALTER TABLE `ercu\_plugin`.`PlateWatchListDetection` MODIFY COLUMN

`siteDetectionCalendarMillis` BIGINT NOT NULL

COMMENT 'The timestamp of the detection by the camera (camera internal calendar time)';

ALTER TABLE `ercu\_plugin`.`PlateWatchListDetection` MODIFY COLUMN

`security` BLOB

COMMENT 'This is the signature computation across: siteId, the plate, createdUtc, siteDetectionUtc, imagePatch, imageOverview, imageFull and the imageCtx fields';

CREATE UNIQUE INDEX `IX\_PlateWatchListDetection\_1` ON

`ercu\_plugin`.`PlateWatchListDetection` (`siteId` ASC, `plate` ASC,

`siteDetectionGpsMillis` ASC);

CREATE TABLE `ercu\_plugin`.`PlateWatchListDetectionImages` (

`id` BIGINT NOT NULL

, `imagePatch` MEDIUMBLOB

, `imageOverview` MEDIUMBLOB

, `imageFull` MEDIUMBLOB

, `imageCtx1` MEDIUMBLOB

, `imageCtx2` MEDIUMBLOB

, `imageCtx3` MEDIUMBLOB

, `imageCtx4` MEDIUMBLOB

, `imageCtx5` MEDIUMBLOB

, `imageCtx6` MEDIUMBLOB

, `imageCtx7` MEDIUMBLOB

, `imageCtx8` MEDIUMBLOB

, PRIMARY KEY (`id`)

, INDEX (`id`)

, CONSTRAINT `FK\_PlateWatchListDetectionImage\_1` FOREIGN KEY (`id`) REFERENCES `ercu\_plugin`.`PlateWatchListDetection` (`id`)

)ENGINE=InnoDB;

# Приложение C – Цифров подпис за запис в таблица PlateWatchListDetection

Всеки запис в таблицата PlateWatchListDetection (като се вземе предвид и неговото копие в таблица PlateWatchListDetectionImages, където има такова) се подписват от вътрешната станция, за да се гарантира целостта на записаните и изпратени данни. Публичното копие на ключа се експортира в предварително съгласуван файл, за да се използва от 3-тата страна (проверка на целостта на данните). Полетата, които се изчисляват в цифровия подпис, са в следния точен ред:

* 1. *id* – масивът от 8 байта, съставящ това число, се изчисляват в процеса на актуализиране на подписа;
  2. *siteId* – масивът от байтове, който представлява кодирането на стринга в поредица от байтове с помощта на група символи, зададени фабрично в платформата, се изчислява в процеса на актуализиране на подписа;
  3. *plate* – масивът от байтове, който представлява кодирането на стринга в поредица от байтове с помощта на група символи, зададени фабрично в платформата, се изчислява в процеса на актуализиране на подписа;
  4. *createdUtc* – масивът от 8 байта, съставящ това число, се изчисляват в процеса на актуализиране на подписа;
  5. *siteDetectionGpsMillis* – масивът от 8 байта, съставящ това число, се изчисляват в процеса на актуализиране на подписа;
  6. *siteDetectionCalendarMillis* – масивът от 8 байта, съставящ това число, се изчисляват в процеса на актуализиране на подписа;
  7. *imagePatch (елемент от таблицата* PlateWatchListDetectionImages*)* –ако не е нула, масивът от байтове, който съставя този голям двоичен обект, се изчислява в процеса на актуализиране на подписа;
  8. *imageOverview (елемент от таблицата* PlateWatchListDetectionImages*)* – ако не е нула, масивът от байтове, който съставя този голям двоичен обект, се изчислява в процеса на актуализиране на подписа;
  9. *imageFull (елемент от таблицата PlateWatchListDetectionImages) – ако не е нула, масивът от байтове, който съставя този голям двоичен обект, се изчислява в процеса на актуализиране на подписа;*
  10. *imageCtx1 до imageCtx8 (елементи от таблицата PlateWatchListDetectionImages) – всички полето imageCtx от 1 до 8, ако не са нули, се изчисляват в подписа по същия начин, като другите изображения: масивът от байтове, който съставя тези големи двоични обекти, се изчислява в процеса на актуализиране на подписа;*

Моля, имайте предвид, че полето *“status”* умишлено не се изчислява в подписа, за да се позволи допълнителното обработване на данните от 3-тата страна.

Важно е да се отбележи също, че при изчисляване на подписа за числата (*id*, *createdUtc* и т.н.) се взима предвид системата за представяне на числа Big Endian (Network Byte Order).

Примерете Java код за това:

**private static** java.security.Signature computeSignature( java.security.Signature signature, PlateWatchListDetection record ) **throws** GeneralSecurityException

{

**if** ( signature != **null** && record != **null** )

{

MEDIUMBLOB

);

/\*

`siteId` VARCHAR(50) NOT NULL

, `plateId` BIGINT

, `createdUtc` BIGINT NOT NULL

, `siteDetectionGpsMillis` BIGINT NOT NULL

, `siteDetectionCalendarMillis` BIGINT NOT NULL

, `status` SMALLINT

, PlateWatchListDetection.`imagePatch` MEDIUMBLOB

, PlateWatchListDetection.`imageOverview`

, PlateWatchListDetection.`imageFull` MEDIUMBLOB

, PlateWatchListDetection.`imageCtx1` MEDIUMBLOB

, PlateWatchListDetection.`imageCtx2` MEDIUMBLOB

, PlateWatchListDetection.`imageCtx3` MEDIUMBLOB

, PlateWatchListDetection.`imageCtx4` MEDIUMBLOB

, PlateWatchListDetection.`imageCtx5` MEDIUMBLOB

, PlateWatchListDetection.`imageCtx6` MEDIUMBLOB

, PlateWatchListDetection.`imageCtx7` MEDIUMBLOB

, PlateWatchListDetection.`imageCtx8` MEDIUMBLOB

\*/

signature.update( CodecUtil.*longToBytes*( record.getId() ) ); signature.update( record.getSiteId().getBytes() ); signature.update( record.getPlate().getPlateText().getBytes()

signature.update( CodecUtil.*longToBytes*(

record.getCreatedUtc() ) );

signature.update( CodecUtil.*longToBytes*( record.getSiteDetectionGpsMillis() ) );

signature.update( CodecUtil.*longToBytes*( record.getSiteDetectionCalendarMillis() ) );

**if**( record.getImages() != **null** )

{

**if**( record.getImages().getImagePatch() != **null** )

{

signature.update( record.getImages().getImagePatch() );

}

**if**( record.getImages().getImageOverview() != **null** )

{

signature.update(

record.getImages().getImageOverview() );

}

**if**( record.getImages().getImageFull() != **null** )

{

signature.update( record.getImages().getImageFull() );

}

**if**( record.getImages().getImageCtx1() != **null** )

{

signature.update( record.getImages().getImageCtx1() );

}

**if**( record.getImages().getImageCtx2() != **null** )

{

signature.update(

record.getImages().getImageCtx2() );

}

**if**( record.getImages().getImageCtx3() != **null** )

{

signature.update( record.getImages().getImageCtx3() );

}

**if**( record.getImages().getImageCtx4() != **null** )

{

signature.update(

record.getImages().getImageCtx4() );

}

**if**( record.getImages().getImageCtx5() != **null** )

{

signature.update( record.getImages().getImageCtx5() );

}

**if**( record.getImages().getImageCtx6() != **null** )

{

signature.update(

record.getImages().getImageCtx6() );

}

**if**( record.getImages().getImageCtx7() != **null** )

{

signature.update( record.getImages().getImageCtx7() );

}

**if**( record.getImages().getImageCtx8() != **null** )

{

signature.update(

record.getImages().getImageCtx8() );

}

}

}

**return** signature;

}

Where the long encoding in CodecUtils. is:

**public static final int** *BYTE\_COUNT\_LONG* = 8;

// Encode methods

**public static byte**[] longToBytes( **long** value )

{

**byte**[] ret = **new byte**[ *BYTE\_COUNT\_LONG* ];

*encodeLong*( value, ret, 0 );

**return** ret;

}

**public static void** encodeLong( **long** encVal, **byte**[] buffer, **int** position )

{

**int** shiftLen = 8 \* *BYTE\_COUNT\_LONG*;

**for**( **int** i = 0; i < *BYTE\_COUNT\_LONG*; i++ )

{

buffer[ position + i ] = 0; shiftLen -= 8;

buffer[ position + i ] |= 0xff & ( encVal >> shiftLen );

}

}

Така голямо десетично число като “123456” (шестнадесетично 0x1E240) би се кодира в следния масив от байтове (шестнадесетични стойности):

[ 0, 0, 0, 0, 0, 1, e2, 40 ]