INSO
16991-1
1st. Edition
Jan.2013

مهوری اسلامی ایران
Islamic Republic of Iran
سازمان ملی استاندارد ایران
Iranian National Standardization Organization

استاندارد ملی ایران
۱۶۹۹۱-۱
چاپ اول
دی ۱۳۹۲

فناوری اطلاعات- شناسایی بسامد رادیویی (RFID) برای مدیریت اقلام: پروتکل داده- قسمت ۱: واسط کاربردی

Information technology - Radio Frequency Identification (RFID) for Item Management: Data Protocol - Part 1: Application Interface

ICS:35.040

به نام خدا

آشنایی با سازمان ملی استاندارد ایران

مؤسسهٔ استاندارد و تحقیقات صنعتی ایران به موجب بند یک مادهٔ۳ قانون اصلاح قوانین و مقررات مؤسسهٔ استاندارد و تحقیقات صنعتی ایران، مصوب بهمن ماه ۱۳۷۱ تنها مرجع رسمی کشور است که وظیفه تعیین، تدوین و نشر استانداردهای ملی (رسمی) ایران را به عهده دارد.

نام موسسه استاندارد و تحقیقات صنعتی ایران به موجب یکصد و پنجاه و دومین جلسه شورای عالی اداری مورخ ۹۰/۶/۲۹ به سازمان ملی استاندارد ایران تغییر و طی نامه شماره ۲۰۶/۳۵۸۳۸ مورخ ۹۰/۷/۲۴ جهت اجرا ابلاغ شده است .

تدوین استاندارد در حوزه های مختلف در کمیسیون های فنی مرکب از کارشناسان سازمان ، صاحب نظران مراکز و مؤسسات علمی، پژوهشی، تولیدی و اقتصادی آگاه و مرتبط انجام می شود وکوششی همگام با مصالح ملی و با توجه به شرایط تولیدی، فناوری و تجاری است که از مشارکت آگاهانه و منصفانهٔ صاحبان حق و نفع، شامل تولیدکنندگان، مصرف‌کنندگان، صادرکنندگان و وارد کنندگان، مراکز علمی و تخصصی، نهادها، سازمان های دولتی و غیر دولتی حاصل می شود . پیش نویس استانداردهای ملی ایران برای نظرخواهی به مراجع ذی نفع و اعضای کمیسیون های فنی مربوط ارسال می‌شود و پس از دریافت نظرها و پیشنهادها در کمیتهٔ ملی مرتبط با آن رشته طرح و در صورت تصویب به عنوان استاندارد ملی (رسمی) ایران چاپ و منتشر می شود.

پیش نویس استانداردهایی که مؤسسات و سازمان های علاقه مند و ذی صلاح نیز با رعایت ضوابط تعیین شده تهیه می کنند درکمیتهٔ ملی طرح و بررسی و درصورت تصویب ، به عنوان استاندارد ملی ایران چاپ و منتشر می شود . بدین ترتیب ، استانداردهایی ملی تلقی می شوند که بر اساس مفاد نوشته شده در استاندارد ملی ایران شمارهٔ ۵ تدوین و در کمیتهٔ ملی استاندارد مربوط که سازمان ملی استاندارد ایران تشکیل می‌دهد به تصویب رسیده باشد.

سازمان ملی استاندارد ایران از اعضای اصلی سازمان بین المللی استاندارد (ISO)[1] ،کمیسیون بین المللی الکتروتکنیک (IEC)[2] و سازمان بین المللی اندازه شناسی قانونی (OIML)[3] است و به عنوان تنها رابط[4] کمیسیون کدکس غذایی (CAC)[5] در کشور فعالیت می کند . در تدوین استانداردهای ملی ایران ضمن توجه به شرایط کلی و نیازمندی های خاص کشور ، از آخرین پیشرفت های علمی ، فنی و صنعتی جهان و استانداردهای بین‌المللی بهره‌گیری می‌شود .

سازمان ملی استاندارد ایران می تواند با رعایت موازین پیش بینی شده در قانون ، برای حمایت از مصرف کنندگان ، حفظ سلامت و ایمنی فردی و عمومی ، حصول اطمینان از کیفیت محصولات و ملاحظات زیست محیطی و اقتصادی ، اجرای بعضی از استانداردهای ملی ایران را برای محصولات تولیدی داخل کشور و/ یا اقلام وارداتی، با تصویب شورای عالی استاندارد، اجباری نماید. سازمان می تواند به منظور حفظ بازارهای بین المللی برای محصولات کشور ، اجرای استاندارد کالاهای صادراتی و درجه‌بندی آن را اجباری نماید . همچنین برای اطمینان بخشیدن به استفاده کنندگان از خدمات سازمان ها و مؤسسات فعال در زمینهٔ مشاوره ، آموزش ، بازرسی ، ممیزی و صدور گواهی سیستم های مدیریت کیفیت و مدیریت زیست‌محیطی ، آزمایشگاه ها و مراکز کالیبراسیون (واسنجی) وسایل سنجش ، سازمان ملی استاندارد ایران این گونه سازمان ها و مؤسسات را بر اساس ضوابط نظام تأیید صلاحیت ایران ارزیابی می کند و در صورت احراز شرایط لازم ، گواهینامهٔ تأیید صلاحیت به آن ها اعطا و بر عملکرد آن ها نظارت می کند . ترویج دستگاه بین المللی یکاها ، کالیبراسیون (واسنجی) وسایل سنجش ، تعیین عیار فلزات گرانبها و انجام تحقیقات کاربردی برای ارتقای سطح استانداردهای ملی ایران از دیگر وظایف این سازمان است .

1- International Organization for Standardization
2 - International Electrotechnical Commission
3- International Organization of Legal Metrology (Organisation Internationale de Metrologie Legale)
4 - Contact point
5 - Codex Alimentarius Commission

کمیسیون فنی تدوین استاندارد

« فناوری اطلاعات- شناسایی بسامد رادیویی (RFID) برای مدیریت اقلام: پروتکل داده- قسمت۱: واسط کاربردی »

رئیس:	**سمت و / یا نمایندگی**
مشرف، بهنوش (فوق لیسانس مهندسی فناوری اطلاعات- شبکه‌های کامپیوتری)	کارشناس استاندارد
دبیر:	
ترابی ، مهرنوش (فوق لیسانس مهندسی فناوری اطلاعات – تجارت الکترونیک)	کارشناس استاندارد
اعضاء: (اسامی به ترتیب حروف الفبا)	
احمدی، محمد (فوق لیسانس مهندسی برق- مخابرات)	کارشناس استاندارد
الماسی، رامین (فوق لیسانس مهندسی کامپیوتر- سخت‌افزار)	کارشناس فناوری اطلاعات اداره کل امور مالیاتی استان هرمزگان
ذاکری، صفورا (فوق لیسانس مهندسی کامپیوتر- نرم‌افزار)	عضو هیات علمی دانشگاه آزاد اسلامی بندرعباس
زمانی ، کرشنا (فوق لیسانس مهندسی فناوری اطلاعات – تجارت الکترونیک)	کارشناس مرکز رایانه دانشگاه مازندران
شایسته، محمد (فوق لیسانس مهندسی کامپیوتر- نرم‌افزار)	عضو هیات علمی دانشگاه آزاد اسلامی بندرعباس

فهرست مندرجات

ادامه فهرست مندرجات

ادامه فهرست مندرجات

ادامه فهرست مندرجات

ادامه فهرست مندرجات

ادامه فهرست مندرجات

ادامه فهرست مندرجات

ادامه فهرست مندرجات

ادامه فهرست مندرجات

ادامه فهرست مندرجات

پیش‌گفتار

استاندارد «فناوری اطلاعات- شناسایی بسامد رادیویی (RFID) برای مدیریت اقلام: پروتکل داده- قسمت ۱: واسط کاربردی» که پیش نویس آن در کمیسیون فنی مربوط، توسط سازمان ملی استاندارد ایران تهیه و تدوین شده و در دویست و نود و هشتمین اجلاسیه کمیته ملی استاندارد رایانه و فرآوری داده مورخ ۹۲/۱۰/۱ مورد تصویب قرار گرفته است اینک به استناد بند یک ماده ۳ قانون اصلاح قوانین و مقررات موسسه استاندارد و تحقیقات صنعتی ایران مصوب بهمن ماه ۱۳۷۱ به عنوان استاندارد ملی منتشر می شود.

برای حفظ همگامی و هماهنگی با تحولات و پیشرفت‌های ملی و جهانی در زمینه صنایع، علوم و خدمات، استانداردهای ملی ایران در مواقع لزوم تجدید نظر خواهند شد و هر گونه پیشنهادی که برای اصلاح و تکمیل این استانداردها ارائه شود، هنگام تجدید نظر در کمیسیون فنی مربوط مورد توجه قرار خواهد گرفت. بنابراین باید همواره از آخرین تجدیدنظر استانداردهای ملی استفاده کرد.

منبع و ماخذی که در تهیه این استاندارد مورد استفاده قرار گرفته است به شرح زیر است:

ISO/IEC 15961-1:2013, Information technology - Radio Frequency Identification (RFID) for Item Management: Data Protocol - Part 1: Application Interface

فناوری اطلاعات- شناسایی بسامد رادیویی (RFID)[1] برای مدیریت اقلام: پروتکل داده- داده- قسمت ۱: واسط کاربردی

۱ هدف و دامنه کاربرد

هدف از تدوین این استاندارد، تعیین واسط انتزاعی بین یک برنامه کاربردی و پردازنده داده می‌باشد و شامل مشخصات و تعریف فرمان‌ها و پاسخ‌های کاربردی است. به داده‌ها و فرمان‌ها اجازه می‌دهد تا در یک روش استاندارد شده‌ای تعیین شوند که این روش، مستقل از واسط هوایی خاص استاندارد ISO/IEC18000 می‌باشد.

این استاندارد:

- راهنماهایی را برای چگونگی نمایش داده‌ها به عنوان اشیا، فراهم می‌کند.
- ساختار Object Identifiers را براساس استاندارد ISO/IEC 9834-1 تعریف می‌کند.
- فرمان‌هایی را تعیین می‌کند که برای انتقال داده بین برچسب RFID و یک برنامه کاربردی پشتیبانی می‌شوند.
- پاسخ‌هایی را تعیین می‌کند که برای انتقال داده‌ها بین برچسب RFID و برنامه کاربردی پشتیبانی می‌شوند.
- هیچ قواعد نحوی انتقال لازم را با استاندارد ISO/IEC15962 تعیین نمی‌کند اما اطلاعات غیرالزامی را در پیوست الف فراهم می‌کند تا یک سازگاری عقب‌گرد را با استاندارد ISO/IEC 15961: 2004 ایجاد کند.

انتظار می‌رود که این استاندارد به عنوان مرجع ایجاد نرم‌افزار مناسب برای کاربردهای خاص و یا برای تجهیزات RFID خاص استفاده شود.

۲ مراجع الزامی

مدارک الزامی زیر حاوی مقرراتی است که در متن این استاندارد ملی ایران به آن‌ها ارجاع داده شده است. بدین ترتیب آن مقررات جزئی از این استاندارد ملی ایران محسوب می‌شود. در صورتی‌که به مدرکی با ذکر تاریخ انتشار ارجاع داده شده باشد، اصلاحیه‌ها و تجدیدنظرهای بعدی آن موردنظر این استاندارد ملی ایران نیست. در مورد مدارکی که بدون ذکر تاریخ انتشار به آن‌ها ارجاع داده شده است، همواره آخرین تجدیدنظر و اصلاحیه‌های بعدی آن‌ها مورد نظر است.

استفاده از مراجع زیر برای این استاندارد الزامی است:

1- Radio Frequency Identification

2-1 ISO/IEC 9834-1, Information technology — Open Systems Interconnection — Procedures for the operation of OSI Registration Authorities: General procedures and top arcs of the International Object Identifier tree (equivalent to ITU-T Recommendation X.660)
2-2 ISO/IEC 15961-3, Information technology — Radio frequency identification (RFID) for item management: Data protocol — Part 3: RFID data constructs
2-3 ISO/IEC 15962:2013, Information technology — Radio frequency identification (RFID) for item management — Data protocol: data encoding rules and logical memory functions
2-4 ISO/IEC 19762-1, Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary — Part 1: General terms relating to AIDC
2-5 ISO/IEC 19762-3, Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary — Part 3: Radio frequency identification (RFID)

۳ اصطلاحات، تعاریف و قراردادها

۳-۱ اصطلاحات و تعاریف

برای اهداف این استاندارد، اصطلاحات و تعاریف موجود در استانداردهای ISO/IEC 1976-1 و ISO/IEC 19762-3 ارائه می‌شوند و موارد زیر به‌کار می‌رود.

۳-۱-۱ برنامه کاربردی[1]

مؤلفه نرم‌افزاری که دستورات را صادر می‌کند و در داخل یک سامانه پاسخ‌هایی را برای این دستورات دریافت می‌کند.

۳-۱-۲ پردازنده داده‌ها[2]

پیاده‌سازی فرآیندهای تعریف شده در استاندارد ISO/IEC 1562، از جمله فشرده‌ساز داده‌ها، شکل‌دهنده، حافظه منطقی و واحد فرمان/ پاسخ.

یادآوری- این در استاندارد ISO/IEC 1591:2004 پردازنده پروتکل داده‌ها[3] نامیده شده است.

۳-۱-۳ شناسه شی نسبی[4]

شناسه شی خاصی که یک شناسه شی ریشه (Root-OID)[5] مشترک (برای اولین قوس‌ها و قوس‌های بعدی) معنی می‌دهد و قوس‌های باقیمانده پس از Root-OID بوسیله Relative-OID تعریف می‌شوند.

1 - Application
2 - Data Processor
3 - Data Protocol Processor
4 - Relative-OID (Relative-Object Identifier)
5 - Root-Object Identifier

۳-۲ قراردادها

بطور قراردادی در استانداردهای بین‌المللی، اعداد طولانی با یک نویسه یا علامت فضای خالی با یک عنوان «هزاران جداکننده»[1] جدا می‌شوند. این استاندارد قراردادی در این استاندارد دنبال نشده است زیرا قوس‌های یک شناسه شی یا یک جداکننده فضای خالی تعریف می‌شوند (برطبق استانداردهای ISO/IEC 8824 و ISO/IEC 8825). چون که نمایش صحیح این قوس‌ها برای این استاندارد ضروری می‌باشد، تمام مقادیر عددی هیچ جداکننده فضای خالی ندارند به جز علامت‌گذاری کردن یک گره بین دو قوس از یک شناسه شی.

۴ رعایت روش‌ها

۴-۱ کلیات

فرمان‌ها یا دستورات و پاسخ‌ها در این استاندارد، در یک قواعد نحوی انتزاعی بیان می‌شوند و کدبندی انتقالی دیگر لازم نمی‌باشد. همین‌طور، رعایت روش‌ها در این استاندارد برای یک سامانه ویژه با کدبندی مناسب برآیند برچسب‌های RFID بر طبق استاندارد ISO/IEC 15962 با استفاده از سامانه نشان داده می‌شود.

متغیرها و فیلدها یا رشته‌ها در داخل هر یک از فرمان‌ها و پاسخ‌ها مشخص می‌کنند که برای ورودی صحیح به پردازنده داده‌ها چه چیزهایی را باید در نظر گرفت تا یک کدبندی معتبر به دست آید. همچنین آن‌ها مشخص می‌کنند که یک برنامه کاربردی چه چیزهایی را باید با دسترسی به یک برچسب RFID برگشت دهد. به دلیل روشی که پروتکل داده‌ها[2] ساخته می‌شوند، این فرمان‌ها و پاسخ‌های مشخص شده در این استاندارد تا حد زیادی جدا از انواع خاص برچسب RFID می‌باشند که تنها از طریق محرک برچسب[3] برای پردازنده داده‌ها شناخته شده هستند. اثر آن این است که استاندارد ISO/IEC 15962 می‌تواند الزامات مطابقت را برای کدبندی معتبر تعیین کند، در حالیکه این استاندارد نمی‌تواند.

زیربندهای زیر عقیده و نظر رعایت روش‌ها را تصریح می‌کنند تا یک کانال ارتباطات داده‌های جامع را بین برنامه کاربردی و برچسب RFID به دست آورند.

۴-۲ رعایت روش‌های برنامه کاربردی

یک برنامه کاربردی از فرمان‌ها و پاسخ‌هایی پشتیبانی می‌کند که برای برنامه کاربردی معنادار می‌باشند. برای هر فرمان مناسب برنامه کاربردی، تمام مؤلفه‌های تشکیل‌دهنده باید در انتقال‌های بین برنامه کاربردی و پردازنده داده‌ها در نظر گرفته شوند.

به‌خصوص، استانداردهای برنامه کاربردی باید متغیرهای مختلف را در فرمان مورد توجه قرار دهند همان‌طور که در بند ۷ تعریف شده است (برای مثال Object- Lock، compact- parameter).

---

1 - Thousands Separator
2 - Data Protocol
3 - Tag Driver

اینها الزاماتی را تعیین می‌کنند که چه چیزی باید بر روی برچسب RFID کدبندی شود و چه فرآیندهای لازمی را پردازنده داده‌ها باید صدا بزند تا یک کدبندی معتبر را به دست آورند.

۴-۳ مطابقت پردازنده داده‌ها

پردازنده داده‌ها، به‌طور مؤثری، پیاده‌سازی استاندارد ISO/IEC 15962 می‌باشد. با توجه به دامنه کاربرد پردازنده داده‌ها (از اینکه مخصوص یک صنعت باشد تا اینکه برای تمام پروتکل داده‌های RFID باشد) متغیرهای مختلف در فرمان‌ها را می‌توان به شیوه‌های مختلفی پردازش کرد (برای مثال، داده‌ها را می‌توان با یک Object-Identifier کامل یا یک Relative-OID شناسایی کرد). این استاندارد هیچ محدودیتی را برای طراحی پردازنده داده‌ها اعمال نمی‌کند، غیر از یک شرایط لازم برای پشتیبانی از تمام قابلیت‌ها و توانایی‌ها که با متغیرهایی در فرمان‌ها مشخص شده است و برای دستیابی به کدبندی مناسب لازم می‌باشند.

۵ مدل پروتکل

۵-۱ مرور کلی

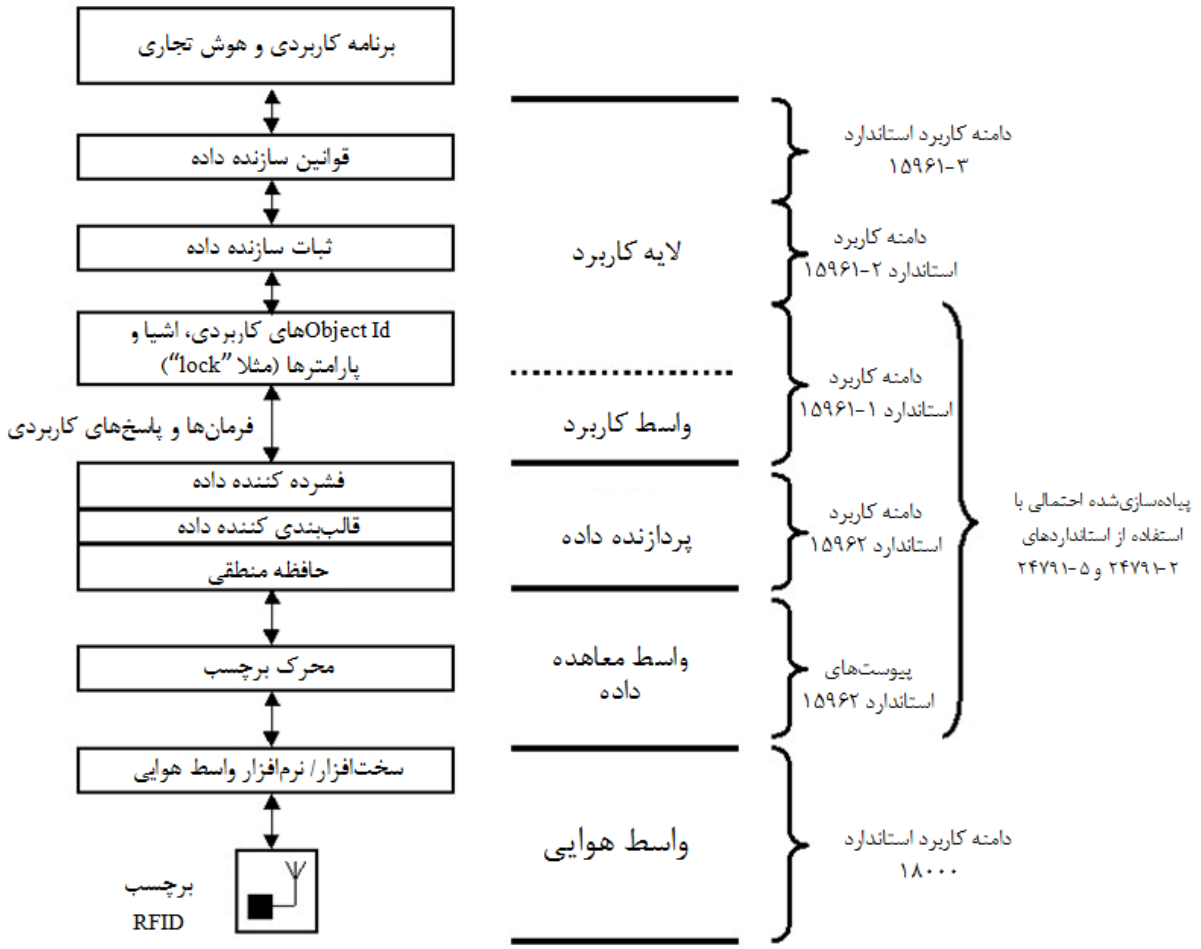
RFID از کدبندی بیت در حافظه برچسب RFID پشتیبانی می‌کند. برخلاف استانداردهای حامل داده‌های دیگر که با استاندارد ISO/IEC JTCI SC 31 تهیه شده‌اند که نیاز به طرح‌های کدبندی دارند و مخصوص فناوری حامل داده‌های شخصی هستند، استاندارد ISO/IEC 18000 تفسیر بیت‌ها یا بیت‌های کدبندی شده بر روی حافظه برچسب RFID را مشخص نمی‌کند. بنابراین چون که یک برچسب RFID در یک سامانه ارتباطات یک بازپخش‌کننده می‌باشند، هر برچسب به کاربرده شده برای مدیریت بخش سامانه‌های باز، باید داده‌هایی داشته باشد که به یک روش ثابت و پایداری کدبندی شده‌اند. عملکرد اصلی این استاندارد، تعیین یک واسط مشترک بین برنامه‌های کاربردی و تحقیق‌کننده RFID می‌باشد. عملکرد اصلی استاندارد ISO/IEC 15962 تعیین قوانین کدبندی مشترک و عملکردهای حافظه منطقی می‌باشد.

برچسب‌های RFID از حافظه الکترونیکی استفاده می‌کنند که می‌توانند ظرفیت داده‌ها را در دوره‌های جدیدی از معرفی محصولات افزایش دهند. تفاوت‌ها در ظرفیت داده‌های هر نوع برچسب RFID، متشابه یا نامتشابه، با پروتکل داده‌هایی که در این دو استاندارد بین‌المللی تعریف شده‌اند، تشخیص داده می‌شوند.

استانداردهای مختلف برنامه کاربردی ممکن است مجموعه داده‌ها یا فرهنگ لغات داده‌های مخصوص خودشان را داشته باشند. هر استاندارد کاربردی مهم برای مدیریت اقلام باید داده‌های خودش را داشته باشد که به یک شیوه واضح به عمل آمده‌اند و از اغتشاش با داده‌های برنامه‌های کاربردی دیگر و حتی با داده‌های سامانه‌های بسته جلوگیری می‌کنند. پروتکل داده‌های مشخص شده در این استانداردها، شناسایی واضح داده‌ها را تضمین می‌کنند.

۵-۲ پروتکل لایه‌ای

لایه‌های پروتکل یک پیاده‌سازی RFID برای مدیریت اقلام در طرح کلی شکل ۱ بصورت نموداری نشان داده شده است- نمودار لایه‌های پروتکل برای پیاده‌سازی RFID برای مدیریت اقلام.



شکل ۱- نمودار لایه‌های پروتکل برای پیاده‌سازی RFID برای مدیریت اقلام

پروتکل داده‌های تعیین شده در استاندارد های ISO/IEC 15961-1، ISO/IEC 15961-2، ISO/IEC 15961-3 و ISO/IEC 15962 مستقل و جدا از فناوری‌های مختلف برچسب RFID می‌باشد که در استاندارد ISO/IEC 18000 تعیین شده است، که در ارتباط با پروتکل‌های واسط هوایی مختلف می‌باشد که بین تحقیق‌کننده و برچسب RFID عمل می‌کنند. این استقلال با پیاده‌سازی استانداردها در سطوح مختلف در سلسله مراتب پروتکل به دست می‌آید. پروتکل داده‌های RFID تعریف شده در این استاندارد در ارتباط با لایه‌های بالاتر می‌باشد همان‌طور که در قسمت زیر توضیح داده شده است.

۵-۲-۱ لایه کاربردی[1]- مطابق با تعریف بخش‌های مختلف استاندارد ISO/IEC 15961

پروتکل داده‌های RFID مشخص می‌کند که چگونه داده‌ها به شکل اشیا نشان داده می‌شوند و هر کدام به‌طور انحصاری با یک شناسه شی شناسایی می‌شود، که برای برنامه کاربردی معنادار هستند و می‌توانند بر روی برچسب RFID کدبندی شوند. استاندارد ISO/IEC 15961-3 قوانین ساخت داده‌ها برای AFI، DSFID، شناسه شی برای شناسه اقلام انحصاری و ساختار شناسه شی برای سایر داده‌های وابسته به اقلام را مشخص می‌کند. این اطمینان می‌دهد که هر قسمت از داده‌ها می‌توانند بطور انحصاری شناسایی شوند و هر دو در دامنه کاربرد یک برنامه کاربردی خاص و بین برنامه‌های کاربردی هستند. هر برنامه کاربردی باید طبق قوانین استاندارد ISO/IEC 15961-2 ثبت شود، بطوریکه و ساختار داده‌ها را می‌توان اعلان و در یک شیوه واضح استفاده کرد.

پروتکل داده‌های RFID در این استاندارد، تابع‌ها و متغیرهایی را برای ساخت و طراحی فرمان‌ها و پاسخ‌های برنامه کاربردی تعریف می‌کند. این همان برنامه‌های کاربردی هستند که می‌توانند تعیین کنند چه داده‌هایی به برچسب RFID و یا از برچسب RFID منتقل شوند و چه داده‌هایی الحاق، به‌روز، بطور انتخابی قفل یا حذف شوند و یا عملکردهای دیگری را بر روی برچسب RFID اجرا کنند.

برای اینکه شرح دهیم چگونه تابع‌ها و متغیرها در یک قالب ساختار یافته مجتمع می‌شوند، چند فرمان و پاسخ با استفاده از یک قواعد نحوی ساخته شده‌اند. این مستقل از برنامه کاربردی میزبان، سیستم عامل و زبان برنامه‌نویسی و همچنین مستقل از ساختارهای فرمان بین تحقیق‌کننده و محرک برچسب می‌باشد. قواعد نحوی انتزاعی به کاربرده شده در این استاندارد، همانند قواعد نحوی به کاربرده شده در استاندارد ISO/IEC 24791-5 می‌باشد و مجتمع‌سازی نزدیک‌تری را با آن استاندارد توانمند می‌کند. نسخه اولیه استاندارد ISO/IEC 15961:2004 فرمان‌هایی را قرار داده است که با استفاده از قواعد نحوی انتزاعی ASN.1 تعریف شده‌اند. برای سازگاری نسخه قبلی، فرمان‌هایی که بصورت ابتدایی با این روش تعریف شده‌اند در این استاندارد در پیوست قرار گرفته‌اند.

همچنین این پروتکل داده‌های RFID، متغیرها و کدهایی را تعریف می‌کند تا پاسخ‌های داده‌هایی را پشتیبانی کند که از یک برچسب RFID خوانده می‌شود، از جمله پیام‌های خطا، که به برنامه کاربردی بر می‌گردند.

قواعد نحوی انتزاعی ممکن است به عنوان یک مبنا برای تهیه فرمان‌ها در زبان‌های مختلف برنامه‌نویسی استفاده شوند تا از عملکرد و متغیرهای فرمان‌های انتزاعی پشتیبانی کنند.

۵-۲-۲ واسط کاربردی[2]- مطابق تعریف استاندارد ISO/IEC 15961-1

واسط کاربردی ممکن است به چند روش مختلف پیاده‌سازی شود که بطور صریح در این استاندارد و استاندارد ISO/IEC 15962 تعریف نمی‌شوند. الزام اصلی، شناسایی هر یک از اشیاء داده‌ها بطور واضح از اشیاء دیگر با استفاده از شناسه‌های شی می‌باشد، حتی قالب‌های مختلف داده‌ها را می‌توانند بر روی یک برچسب RFID

1 - Application Layer
2 - Application Interface

همانند ترکیب کنند. همچنین واسط کاربردی باید متغیرهای فرمان و پاسخ را واضح تعریف کند بطوریکه آن‌ها بتوانند با داده‌های یک شبکه سیمی یا بی‌سیم مشابه ترکیب شوند.

یک کلاس مهم پیاده‌سازی، که با عنوان فرآیند straight-through، توصیف شده است، در جایی مناسب است که تابع‌ها ومتغیرهای به کار رفته برای ساخت فرمان‌ها و متغیرها و کدها برای ساخت پاسخ‌ها نیز به کار رفته است، همان‌طور که در این استاندارد تعیین شده است، که بطور مستقیم ورودی فرآیندهای کدبندی استاندارد ISO/IEC 15962 می‌باشند. این ورودی می‌تواند از صفحه‌های کامپیوتر یا فرم‌ها، یا انتقال‌های مستقیم‌تر از سامانه‌های میزبان باشد. مزیت این فرآیند این است که از تشکیل کدبندی انتقالی جلوگیری می‌کند (به قسمت پایین مراجعه کنید)، اما به پیوستگی سخت‌تری در الزامات عملکردی فرمان‌ها و پاسخ‌ها نیاز دارد. این استاندارد هیچ محدودیتی را در فرآیند مخصوص واسط کاربردی انتخاب شده اعمال نمی‌کند، مگر این شرط که با قوانین کدبندی استاندارد ISO/IEC 15962 مجتمع شود.

یک فرآیند دیگر، مطابق با اولین ویرایش استاندارد ISO/IEC 15961، استفاده از قواعد نحوی انتزاعی برای تعریف فرمان‌ها و پاسخ‌ها در یک روش ساختاریافته، پایدار و قابل رسیدگی و همچنین تولید کدبندی انتقالی که جریان بیت منتقل شده بین فرآیندهای این استاندارد بین‌المللی و فرآیندهای استاندارد ISO/IEC 15962 را تعریف می‌کند، می‌باشد.

از هر روشی که استفاده شود، قوانین کدبندی استاندارد ISO/IEC 15962 باید دنبال شود و کدبندی بر روی برچسب RFID باید با تمام متغیرهای فرمان‌های تعیین شده در این استاندارد مطابق باشد.

۵-۲-۳ پردازش پروتکل داده‌ها[1]- مطابق تعریف استاندارد ISO/IEC 15962

پروتکل داده‌های RFID تعیین می‌کند که چه مقدار از داده‌ها بر روی برچسب RFID کدبندی، فشرده و قالب‌بندی می‌شود و چگونه این داده‌ها از برچسب RFID بازیابی می‌شود تا در برنامه کاربردی معنادار باشد.

این پروتکل داده‌های RFID برای مجموعه‌ای از طرح‌هایی تهیه می‌شود که داده‌ها را فشرده می‌کنند تا در فضای حافظه بهتر استفاده شوند.

همچنین این پروتکل داده‌های RFID، از فرمت‌های مختلف ذخیره‌سازی پشتیبانی می‌کند تا بتوانند از حافظه بهتر استفاده کنند و دسترسی کاراتر به رویه‌ها داشته باشند.

۵-۲-۴ واسطه پروتکل داده‌ها[2]- مطابق تعریف استاندارد ISO/IEC 15962

هر استاندارد پروتکل واسط هوایی موجود در استاندارد ISO/IEC 18000، قوانین مخصوص خودش را برای تعریف فرمان‌ها و پاسخ‌ها دارد. همچنین یک پروتکل واسط هوایی می‌تواند از معماری‌های سامانه برچسب مختلف با اندازه‌های حافظه مختلف پشتیبانی کند و ممکن است فرمان‌های اختیاری را پشتیبانی کند. با این قوانین، پروتکل داده‌ها یک سازوکاری را برای واسط، از طریق محرک‌های برچسب خاص فراهم می‌کند. اینها

1 - Data Protocol Processing
2 - Data Protocol Interface

امکاناتی را فراهم می‌کنند تا فرمان‌های کاربردی اصلی و پاسخ‌های این استاندارد، جدا از پروتکل واسط هوایی و معماری سامانه برچسب خاص به کار برده شوند.
مؤلفه محرک برچسب پروتکل داده‌ها، قانون نگاشت را از فرآیندهای کلی به الزامات برچسب خاص فراهم می‌کند. این قواعد نگاشت، برای نوشتن و خواندن داده‌ها استفاده می‌شوند.
محرک‌های برچسب اضافی را می‌توان به عنوان پروتکل‌های واسط هوایی جدید تعیین کرد که در مجموعه استانداردهای ISO/IEC 18000 معرفی شده‌اند.

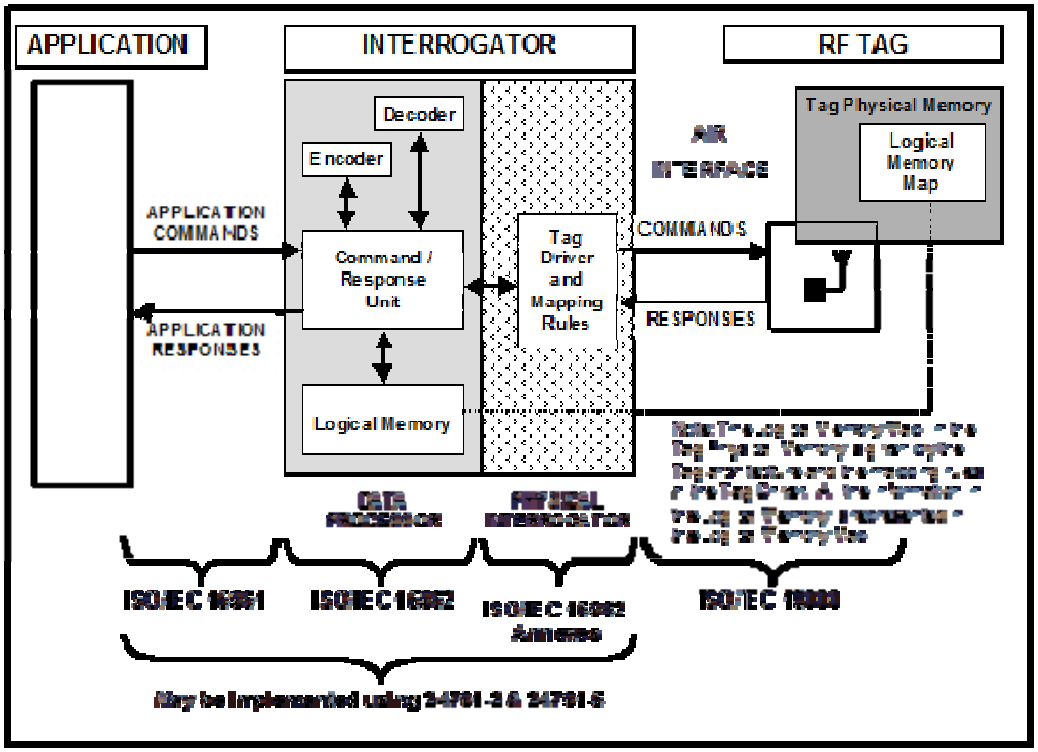
۵-۳ پیکربندی‌های پیاده‌سازی انعطاف‌پذیر

این پروتکل داده‌های RFID، ارتباطات سطح کاربردی و قواعد سطح تحقیق‌کننده برچسب RFID را برای کدبندی، فشرده‌سازی و قالب‌های ذخیره‌سازی تعیین می‌کند. این پروتکل ممکن است همراه با موارد زیر پیاده‌سازی شود:

- با استاندارد ISO/IEC 15962 که در معماری زیرساخت سامانه نرم‌افزاری که در استاندارد ISO/IEC 24791 تعریف شده است، گنجانده شده است. این روش پیشنهادی برای هر برنامه کاربردی شبکه‌سازی شده می‌باشد.
- با این استاندارد و استاندارد ISO/IEC 15962 که در نرم‌افزار مستقل یا افزاره‌هایی که بعنوان خروجیش کدبندی و / یا کدگشایی تطبیقی با پاسخ‌های که با پاسخ‌های این استاندارد مطابقت دارد، گنجانده شده است.

۵-۴ فرآیندهای کارکردی- پیاده‌سازی تحقیق‌کننده

فرآیندهای کارکردی مختلفی وجود دارند که باید برای نوشتن داده‌ها بر روی یک برچسب RFID و یا خواندن داده‌ها از آن، اتفاق بیافتند. شکل ۲- توابع منطقی و واسطهای استاندارد ISO/IEC 15962 به همراه سایرمؤلفه‌های سامانه RFID- یک نموداری از یک نمونه پیاده‌سازی را نشان می‌دهد که در آنجا پردازش پروتکل داده‌ها در قسمت تحقیق‌کننده می‌ماند. این شرح و توضیح تهیه می‌شود تا به شناخت و درک فرآیندها کمک کند و هرچند به پیاده‌سازی نیز کمک می‌کند، بسیاری از توضیحات ممکن است موافق با این پروتکل داده‌ها باشند.



شکل ۲- توابع منطقی و واسط‌های استاندارد ISO/IEC 15962 به همراه سایرمؤلفه‌های سامانه RFID

۵-۴-۱ فرآیندهای کارکردی- واسط کاربردی

جریان‌های داده‌ها بین برنامه کاربردی و پردازنده داده‌ها مطابق این استاندارد شکل می‌گیرند و فشرده نیستند. بنابراین سامانه‌های زیادی ایجاد شده‌اند که در آنجا داده‌ها طبق توافق (مثلا با قواعد نحوی مربوطه به کد میله‌ای) شکل می‌گیرند. بنابراین منطقی است که پودمان[1]های واسط را در جریان داده‌ها قرار دهید تا از قالب‌های کاربردی موجود و به قالب‌های کاربردی موجود تبدیل کنید.

یادآوری- توجه دقیق باید تا حدی داده شود که سامانه‌های مستقر باید نسبت به مزیت‌های ممکنی که از انتخاب پروتکل داده‌ها به دست می‌آورند پشتیبانی شوند که در این استاندارد و استاندارد ISO/IEC 15962 تعیین شده‌اند. زیرا که این پروتکل در مورد ویژگی‌های RFID، از قبیل خواندن/ نوشتن انتخابی و توانایی قفل داده‌ها، ایجاد شده است. پروتکل‌های قدیمی‌تر بعید است که از این ویژگی‌ها پشتیبانی کنند.

۵-۴-۲ فرآیندهای کارکردی- تحقیق‌کننده

در فرآیند شرح داده شده در شکل ۲، تحقیق‌کننده، پودمانی است که در آن تمام پردازش اصلی پروتکل داده‌ها روی می‌دهد و یک واسط در برچسب RFID وجود دارد. پیاده‌سازی‌ها مختلف ممکن است بعضی از تابع‌هایی را جدا کند که در قسمت زیر شرح داده شده اند و واسطی را بین برنامه کاربردی و تحقیق‌کننده فیزیکی دارند.

1 - Module

۵-۴-۲-۱ پردازنده داده‌ها

پردازنده داده‌ها تمام پردازش‌هایی را فراهم می‌کند که مطابق با استاندارد ISO/IEC 15962 است و برای جابجایی داده‌های کاربردی لازم می‌باشد. شامل مؤلفه‌های زیر است که تمام آن‌ها در قسمت زیر کامل‌تر توصیف می‌شوند:

واحد فرمان/پاسخ[۱]، حافظه منطقی[۲]، کدگذار[۳] (که تابع فشرده‌ساز داده[۴] و شکل‌دهنده[۵] را پشتیبانی می‌کند) و کدگشا[۶] (که توابع معکوس کدگذار را پشتیبانی می‌کند). پردازنده داده بطور فیزیکی می‌تواند در هر جایی بین نرم‌افزار کاربردی و محرک برچسب قرار گیرند اما باید تمام مؤلفه‌ها را داشته باشد. بعضی یا تمام توابع پردازنده داده ممکن است با استفاده از فرآیندهایی اجرا شوند که در استانداردهای ISO/IEC 24791-2 و ISO/IEC 24791-5 تعریف شده‌اند.

۵-۴-۲-۱-۱ واحد فرمان/ پاسخ

واحد فرمان/پاسخ، فرمان‌های کاربردی را از برنامه کاربردی در قالب تعیین شده در این استاندارد دریافت می‌کند و بر روی این فرمان‌ها در جاهای مناسب، عمل می‌کند و به کدهای فرمان سطح پایین برچسب RFID مشخص تبدیل می‌شوند.

مثال: یک فرمان کاربردی نوشتن Data Object {name} به برنامه کاربردی مربوط می‌شود. پروتکل داده‌ها این را تشخیص می‌دهد و می‌تواند داده‌ها را بر روی حافظه منطقی در پردازنده داده قالب‌بندی کند. اطلاعات برچسب ویژه RFID لازم است تا پارامترهای نگاشت حافظه منطقی[۷] (مثلا تعداد بایت‌ها، خواه از یک فهرست راهنما استفاده شود، غیره) را بر روی برچسب RFID قرار دهند. محرک برچسب، فرمان برنامه کاربردی را به فرمان مشخص‌شده برچسب، تبدیل می‌کند.

از این مثال دیده می‌شود که یک حد منطقی جدایی بین پردازنده داده و محرک برچسب وجود دارد.

۵-۴-۲-۱-۲ حافظه منطقی

حافظه منطقی آرایه‌ای از بایت‌های پیوسته حافظه است که همانند یک نمونه نرم‌افزار مشترک از نگاشت حافظه منطقی، در حافظه کاربر برچسب RFID عمل می‌کند که در آن شناسه شی و اشیا داده در بایت‌ها نگاشت می‌شوند. حافظه منطقی چند پارامتر از برچسب حقیقی RFID را مورد توجه قرار می‌دهد، برای مثال، اندازه بلوک، تعداد بلوک‌ها و قالب ذخیره‌سازی. حافظه منطقی هر ساختار معماری برچسب که با جزئیات است، را نادیده می‌گیرد.

---

1 - Command/Response Unit
2 - Logical Memory
3 - Encoder
4 - Data Compactor
5 - Formatter
6 - Decoder
7 - Logical Memory Map

کاربرد حافظه منطقی بدین‌معناست که یک برنامه کاربردی می‌تواند واسط با یک برچسب RFID موافق با برنامه کاربردی[1] باشد، اما آن برچسب‌های RFID اختصاصی می‌توانند ظرفیت‌های حافظه و ساختارهای معماری به‌طورکامل متفاوتی داشته باشند. بنابراین، پیاده‌سازی، از توسعه‌های فناوری جدید، که در چارچوب استاندارد ISO/IEC 18000 مجاز بوده است، بهره‌مند می‌شود، از قبیل ظرفیت بیشتر یا برچسب‌های RFID با دسترسی سریع‌تر، بدون تغییر برنامه کاربردی.

۵-۴-۲-۱-۳ رمزگذار

رمزگذار فرآیند نوشتن داده‌ها را در داخل پردازش‌های کارکردی کنترل می‌کند که با پودمان فشرده‌ساز داده و پودمان شکل‌دهنده اجرا شده است.

۵-۴-۲-۱-۴ فشرده‌ساز داده‌ها

فشرده‌ساز داده‌ها قواعد استاندارد فشرده‌سازی را تهیه می‌کند تا تعداد بایت‌های ذخیره شده بر روی برچسب RFID را کاهش و در واسط هوایی انتقال دهد. برای مثال، داده عددی برای برنامه کاربردی براساس هشت‌تایی مجموعه نویسه کدشده می‌باشد، اما به شکل فشرده بر روی حافظه برچسب RFID می‌تواند کدبندی شود.

۵-۴-۲-۱-۵ شکل‌دهنده

شکل‌دهنده فرآیندهایی را فراهم می‌کند تا شناسه داده و شی (داده)identifier را در قالب مناسب و کارآمد برای ذخیره‌سازی بر روی حافظه منطقی قرار دهد.

یادآوری- نگاشت فیزیکی بیت‌ها برای مطابقت با معماری برچسب RFID، براساس اطلاعات فراهم شده توسط محرک برچسپ، اجرا می‌شود.

۵-۴-۲-۱-۶ رمزگشا

رمزگشا فرآیند خواندن و تفسیر داده‌ها را در داخل پردازش‌های کارکردی کنترل می‌کند که با استفاده از Data De-compactor Module و De-formatter Module اجرا شده است.

۵-۴-۲-۲ محرک برچسب

محرک برچسپ دو کارکرد مهم را فراهم می‌کند:

- قواعد نگاشت بر روی ساختار برچسب RFID را تهیه می‌کند تا محتوای حافظه منطقی در پردازنده داده با نگاشت حافظه RFID در زمان استفاده مبادله شوند.
- امکاناتی را فراهم می‌کند که فرمان‌های برنامه کاربردی این پروتکل داده‌ها را قبول می‌کند و آن‌ها را به یک قالبی تبدیل می‌کند که موجب فراخوانی کدهای فرمان می‌شود که با برچسب ویژه RFID پشتیبانی شده است.

1 - Application-Compliant RFID tag

برای مثال یک فرمان با دستور برنامه کاربردی write Data Object{name} ممکن است منجر به فرمان نوشتن برچسب RFID شود (بلوک #، داده‌ها).

توصیف محرک برچسب برای برچسب‌های ویژه FRID در پیوست‌های استاندارد ISO/IEC 15962 تهیه شده است. به منظور استاندارد ISO/IEC 15962، همان‌طور که در یکی از قسمتهای مناسب استاندارد ISO/IEC 18000 تعیین شده است، یک محرک برچسب در یک نوع واسط هوایی ویژه برچسب RFID منحصر به فرد است. این یک نمونه منطقی است؛ پیاده‌سازی فیزیکی می‌تواند ویژگی‌های محرک‌های برچسب منطقی مختلف را با هم ترکیب کند. یک تحقیق‌کننده ممکن است از یک یا بسیاری از محرک‌های برچسب پشتیبانی کند.

۵-۴-۲-۳ سازوکارهای انتقال

سازوکارهای انتقال برای انتقال داده‌ها، فرمان‌ها و پاسخ‌ها بین پردازنده داده (یعنی پیاده‌سازی این استاندارد) و برچسب RFID، باید از طریق تحقیق‌کننده انجام گیرد و یا در کارکردهای تحقیق‌کننده ترکیب شود.

۵-۴-۳ برچسب RFID

اگرچه برچسب RFID فراتر از هدف و دامنه کاربرد این استاندارد می‌باشد، این برچسب در شکل ۲- توابع منطقی و واسط‌های استاندارد ISO/IEC 15962 به همراه سایرمؤلفه‌های سامانه RFID - نشان داده می‌شود تا جریان داده‌ها و فرمان‌ها را کامل کنند. در برچسب RFID، نگاشت حافظه منطقی، تمام داده‌ها را در حافظه منطقی از پردازنده داده نشان می‌دهد که به یک ساختار موقعیتی تبدیل (یا نگاشت) می‌شود که با قواعد نگاشت در محرک برچسب و معماری برچسب RFID تعیین شده‌اند.

۵-۵ استاندارد ISO/IEC 15962 و پردازنده داده‌ها

استاندارد ISO/IEC 15962 تمام قواعد برای کدبندی داده‌ها را بر روی یک برچسب RFID تعریف می‌کند. پیاده‌سازی این قواعد در این استاندارد توصیف می‌شود که همانند پردازنده داده می‌باشد. همان‌طور که در بند ۵-۴ شرح داده شد، پرداش‌های مختلف قبول می‌کنند که کدبندی موفقیت‌آمیزی را به دست آورند. این قواعد که بایت‌های کدبندی شده را به دست می‌آورند در یک عملیات خواندن «خود-اعلانی[1]» هستند و هیچ دانش قبلی یا مستقلی از کدبندی‌ها بر روی برچسب ندارند. این به پروتکل داده اجازه می‌دهد که در سامانه‌های باز استفاده شود که در آنجا سازمانی که داده‌ها را بر روی برچسب RFID کدبندی می‌کند ممکن است به‌طورکامل ناآگاه باشد از اینکه سازمان چه چیزی را ممکن است از داده‌های برچسب RFID بخواند. این به دست می‌آید با:

- داشتن قواعد تعریف شده بطور صریح برای فشرده‌سازی داده‌هایی که جدا از برنامه کاربردی و نوع برچسب RFID به کار برده می‌شوند.
- تعریف قواعد ساخت اصولی برای کدبندی داده‌هایی که تا درجه زیادی توسط یک ویژگی قابل انتخاب بوسیله کاربر[2] به نام Access-Method تعیین شده‌اند.

1 - Self-declaring
2 - User-selectable Feature

- داشتن قواعد ساخت دقیق در داخل هر Access-Method، از جمله یک قواعد نحوی تعریف شده بطور صریح که به سامانه‌های خواندن اجازه می‌دهد که بطور انتخابی داده‌های لازم برای برنامه کاربردی مشخص کنند.

بند ۷-۲-۴ یک توصیف و یک تعریف کافی برای یک مدیر برنامه کاربردی فراهم می‌کند تا مزیت‌های چهار Access-Methods را ارزیابی کنید که در حال حاضر پشتیبانی شده‌اند. پردازش مربوطه در استاندارد ISO/IEC 15962 به‌طورکامل مشخص شده است.

۶ ارائه قراردادها

۶-۱ ارائه فرمان‌ها، پاسخ‌ها و متغیرها

۶-۱-۱ فرمان‌ها و پاسخ‌ها

فرمان‌ها و پاسخ‌ها در یک جعبه با حاشیه دو خط تعریف می‌شوند همان‌طور که در شکل ۳ - شکل جعبه برای فرمان‌ها و پاسخ‌ها- نشان داده می‌شود.

شکل۳- شکل جعبه برای فرمان‌ها و پاسخ‌ها

هر فرمان، یا پاسخ، شامل یک فهرست مرتب شده از فیلدها و متغیرها می‌باشد. نام فیلد/متغیر به شکل **برجسته**[1] نشان داده می‌شوند: **Argument-Name** زمانی که لازم باشد، نام فیلد/متغیر همراه با نوع داده‌ها و یک توصیف مختصر می‌باشد. فیلدها با مقادیری که در یک زیرمجموعه‌ای از دامنه انواع داده‌های خودشان محدود هستند مقادیر ممکن و قانونی خودشان را دارند که به صورت *کج*[2]، زیر نام رشته نشان داده شده است.

یادآوری- این موارد، هنگامی‌که در قواعد نحوی انتزاعی ASN.1 (همانطور که در استاندارد ISO/IEC 15961: 2004 می‌باشد) نمایش داده می‌شوند، در فرمان‌ها و پاسخ‌های اصلی به کار برده نمی‌شود.

۶-۱-۲ متغیرها

متغیرهای پیچیده‌تر که شامل زیرمتغیر هستند، درون جعبه‌هایی با حاشیه یک خط تعریف می‌شوند مانند شکل ۴- شکل جعبه برای متغیرها.

شکل ۴- شکل جعبه برای متغیرها

1 - Bold
2 - Italic

۶-۱-۳ انواع داده‌ها

انواع داده‌های زیر در فرمان‌ها و پاسخ‌ها استفاده می‌شوند:

- BOOLEAN: یک متغیر که می‌تواند مقادیر TRUE یا FALSE داشته باشد.
- BIT STRING: یک ترتیبی از بیت‌ها.
- BYTE: یک عدد صحیح با مقادیر بین ۰ تا ۲۵۵، به‌طورمعمول به عنوان یک مقدار شانزده شانزدهی OO_{16} تا FF_{16} بیان می‌شود.
- BYTE STRING: یک ترتیبی از بایت‌ها (معادل OCTET STRING)
- EBV-8: یک روش دودویی برای کدبندی اعداد با اندازه متغیر در فیلد همانند با استفاده از یک بیت علامت پیشتاز قبل از یک مقدار ۷ بیتی. مؤلفه EBV-8 نهایی، با یک 0 شروع می‌شود، تمام مؤلفه‌های قبلی با یک ۱ شروع می‌شوند. برای مثال، $69_{10} = 01000101$ ، در حالی که $369_{10} = 101110001_2$ که نمایش EBV-8 آن **0**1110001 **1**00000 می‌باشد (یعنی در رشته‌های ۷- بیتی جدا شده است و سپس این علامت به عنوان یک پیشوند اضافه شده است). تنها شرط لازم برای استفاده از یک کد EBC-8 در یک فرمان این است که این نوع مقدار در یک پاسخ واسط هوایی به کجا بر می‌گردد.
- HEXADECIMAL ADDRESS: یک موقعیت (در حافظه یک برچسب RFID)، که به عنوان یک مقدار شانزده شانزدهی بیان شده است.
- INTEGER: یک عدد صحیح می‌تواند هر عددی را بگیرد. در متن این استاندارد این مقادیر همه مثبت هستند.
- OBJECT IDENTIFIER: یک شناسه شی است که در بند۶-۲-۱ تعریف شده است.

۶-۲ ارائه شناسه شی در واسط کاربردی

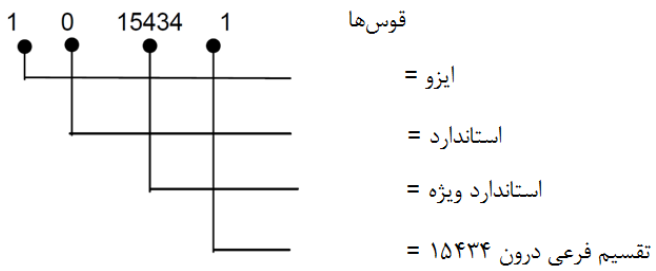
۶-۲-۱ ساختار شناسه شی در استاندارد ISO/IEC 8824-1

این استاندارد از نوع شناسه شی[1] (مطابق تعریف در استاندارد ISO/IEC 8824-1) همراه با تخصیص شناسه‌های که مطابق تعریف در استاندارد ISO/IEC 9834-1 است، استفاده می‌کند. از یک درخت ثبت استفاده می‌کند که یک گره ریشه ضمنی مشترک دارد (مطابق با استاندارد ISO/IEC 9834-1) و یک سری قوس‌ها از هر گره دارد؛ قوس‌های جدید در شرایط لازم اضافه شده‌اند تا یک شی خاص را تعریف کنند. بنابراین این نهاد مسئول یک گره ویژه است که:

- یک مجموعه تعریف شده‌ای از قوس‌ها را دارد تا خودش را شناسایی کند.
- می‌تواند تخصیص قوس‌ها را زیر گره آن، جدا از نهادهای دیگر مدیریت و کنترل کند.
- از یکتایی از تمام قوس‌های دیگر در درخت ثبت مطمئن است.

1 - Object Identifier

مثال:



تنها قوس‌های بالایی برای تمام شناسه‌های شی مجاز هستند که در جدول ۱- قوس‌های بالایی شناسه شی- نشان داده می‌شوند.

جدول ۱- قوس‌های بالایی شناسه شی

نام قوس شناسه	مقدار عددی
itu-t	۰
iso	۱
joint-iso-iut-t	۲

یادآوری- همه استانداردهای ISO/IEC، از جمله این استاندارد، در زیر قوس بالایی Iso، شناسه‌های شی دارند.

قوس دوم توسط سازمان مربوطه مدیریت می‌شود که برای قوس بالا نامیده شده است. فهرست فعلی از قوس‌های بالا و دوم در قسمت سوم استاندارد ISO/IEC 15961 ارائه می‌شود.

قوس سوم با سامانه یا نهادی کنترل می‌شود که برای قوس دوم تعریف شده است؛ گاهی اوقات این یک اداره ثبت‌نام[1] است. ساختار سلسله مراتبی ادامه دارد تا اینکه شی بطور منحصر به فردی شناسایی می‌شود.

دستورالعمل نام‌گذاری شناسه‌های شی تضمین می‌کند که هر شی در داخل قوس «والد» خود بی‌نظیر و منحصر به فرد است و اینکه هر قوس والد در داخل سطح قبلی خود منحصر به فرد است، به ۳ قوس بالا بر می‌گردد.

یادآوری- این ساختار، شناسه‌های شی را از قلمروهای مختلف توانمند می‌کند (بعنوان مثال، سامانه‌های باز و بسته) تا بدون ابهام بر روی یک حافظه برچسب RFID کدبندی شوند.

سه شکل از شناسه شی با پروتکل داده RFID استفاده می‌شوند:

- **Object-identifier**: این ساختار کامل برای ارتباطات بین برنامه کاربردی و پردازنده داده استفاده می‌شود که بوسیله این استاندارد تعریف شده است. ساختار کامل برای استفاده در استاندارد

1 - Registration Authority

ISO/IEC 15962 جایی‌که مجموعه‌ای از شناسه‌های شی که بر روی برچسب RFID کدبندی می‌شوند، قوس‌های سطح بالاتر مختلفی دارند و یا جایی که یک سامانه کلی در محل وجود دارد، مناسب است.

- **Relative-OID**: این ساختار به همراه **Root-OID** (به قسمت پایین مراجعه شود) برای ارتباطات بین برنامه کاربردی و پردازنده داده ستفاده می‌شود که در این استاندارد تعریف شده است. این ساختارها در موقعیت‌هایی استفاده می‌شوند که یک ریشه مشترک در مجموعه‌ای از شناسه‌های شی به کاربرده می‌شود تا بر روی برچسب RFID کدبندی شود. برای مثال، اگر تمام شناسه‌های شی، ریشه مشترک 12 15961 10 داشته باشند، در صورتی‌که نیازی نباشد که این **Root-OID** مشترک برای هر شناسه‌های شی کدبندی شود، فضای کدبندی را می‌توان بر روی برچسب RFID ذخیره کرد. **Relative-OID** یک پسوند برای **Root-OID** مشترک است که به روش دیگری، کدبندی یا اعلان می‌شود.

یادآوری- به جز در شکل‌های انتزاعی فرمان و پاسخ (به بند ۱۰ مراجعه کنید) و مکان‌هایی که آنجا تمایز مهم و حیاتی است، جمله[1] شناسه شی نیز برای Root-OID استفاده می‌شود.

- **Root-OID**: **Root-OID** بخش مشترکی از یک مجموعه شناسه‌های شی کدبندی شده می‌باشد. همانند یک پیشوند مشترک در مقادیر **Relative-OID** عمل می‌کند که بر روی برچسب RFID کدبندی شده است. این ساختار در برنامه‌های کاربردی اهمیت خاصی دارد که به چند نوع داده از یک فرهنگ لغت داده‌های مشترک نیاز می‌باشد تا بر روی یک برچسب RFID کدبندی شود.

Root-OID بطور صریح کدبندی می‌شود و یا به طریق دیگری که برطبق قوانین کدبندی استاندارد ISO/IEC 15962 است، اعلان می‌شود.

۶-۲-۲ ارائه Object-Identifier در شیوه نگارش استاندارد ISO/IEC 8824-1

زمانی که **Object-Identifier** در شیوه نگارش استاندارد ISO/IEC 8824-1 نشان داده می‌شود، فضاها بین هر قوس همانند زیر درج می‌شوند:

1 0 15961 12 1

یادآوری- زیربند ۳۱-۱۲ **از** استاندارد ISO/IEC 8824-1: 2003 مجموعه‌ای از نوشتارهای معادل را نشان می‌دهد از جمله روشی که در بالا نشان داده شده است. نمونه رسمی‌تر از این در ASN: 1 این است: {iso(1) standard(0) rfid-data-protocol(15961) iata(12) baggage-id(1)}.

1 - Term

۶-۲-۳ ارائه Object-Identifier به عنوان یک نام منبع یکسان (URN)[1]

Object-Identifier نیز ممکن است در URN در فرمت یا الگوی زیر نشان داده شود که براساس استاندارد IETF RFC3061 با علامت نقطه اعشار بین هر قوس می‌باشد:

urn:oid:1.0.15961.12.1

۶-۳ نوشتار بایت

۶-۳-۱ بایت: واحد اصلی برای کدبندی ۸بیتی

این استاندارد، داده‌های دودویی، ۷ بیتی، ۸ بیتی و داده‌های کاربر را که ممکن است در هر نویسه بیشتر از ۸ بیت باشد. واحد مشترک کدبندی بایت ۸ بیتی است (همچنین به عنوان هشتایی شناخته شده است). داده‌های دودویی باید با بیت‌های صفر مقدم و پیشتاز پوشش داده شوند تا اینکه مقدار دودویی در ردیف هشتایی باشد، داده‌های ۷ بیتی باید به صورت بایت‌ها با ۸ بیت نشان داده شوند (به زیربند ۶-۳-۲ مراجعه کنید) که بیت هشتم مقدار صفر خواهد داشت. داده‌های بیشتر از ۸ بیت باید در چندین بایت کدبندی شوند.

یک بایت به صورت ۲ نویسه شانزده شانزدهی با مقادیر0-9,A-F نشان داده می‌شود.

۶-۳-۲ ترتیب بیت

در هر بایت، مهمترین بیت، بیت ۸ است و کم‌اهمیت‌ترین بیت، بیت ۱ است. همچنین وزن تخصیص داده شده به هر بیت در جدول ۲ - ترتیب بیت در بایت ۸ بیتی - نشان داده می‌شود.

جدول ۲- ترتیب بیت در بایت ۸ بیتی

مقدار بیت	بیت ۱	بیت ۲	بیت ۳	بیت ۴	بیت ۵	بیت ۶	بیت ۷	بیت ۸
وزن	۱	۲	۴	۸	۱۶	۳۲	۶۴	۱۲۸

۶-۳-۳ تبدیل بایت

مقدار ۸ بیت به دو نویسه شانزده شانزدهی با بیت ۸، بیت ۷، بیت ۶ و بیت ۵ تبدیل می‌شود که به ترتیب وزن‌های ۸، ۴، ۲ و ۱ دارند تا اولین نویسه شانزده شانزدهی را تعریف کنند. بیت ۴، بیت ۳، بیت ۲ و بیت ۱ وزن‌های ۸، ۴، ۲، ۱ را به ترتیب حفظ می‌کند تا نویسه شانزده شانزدهی دوم را تعریف کنند.

۷ پردازش فرمان‌ها و پاسخ‌های برنامه کاربردی

۷-۱ کلیات

از زمان انتشار اولین ویرایش استانداردهای ISO/IEC 15961 و ISO/IEC 15962، معلوم شد که چندین پیکربندی افزاره نیازی به کدبندی انتقال ندارد تا از کارکردهای پروتکل داده به درستی پشتیبانی کنند. این پیکربندی‌های افزاره مشخصات متشابهی دارند بطوری که می‌توانند تمام فرآیندها یا پردازه‌های داخلی را

1 - Uniform Resource Name

پشتیبانی کنند (مثال‌ها شامل کدکننده‌های- چاپگر، خواننده‌های RFID دستی، و بسته‌های نرم‌افزاری می‌باشند)، اما پیکربندی‌های افزاره دیگر می‌توانند مشخصات متشابهی داشته باشند. این مشخصات شامل یک فرآیند کدبندی کامل هستند که از قابلیت و توانایی پردازه‌های کدبندی و فرمان برنامه کاربردی بر روی حافظه منطقی و یا پردازه‌های رمزگشایی از حافظه منطقی در پاسخ‌های برنامه کاربردی پشتیبانی می‌کند.
در این افزاره‌ها، کدبندی انتقالی باید ساخته شود و فوری رمزگشایی شود و دور انداخته شود، در نتیجه دو پردازش غیرضروری حاصل می‌شود که به زمان پردازش و پیچیدگی آن می‌افزاید، به عنوان بخشی از تجدیدنظر استاندارد ISO/IEC 15961 در یک استاندارد چندبخشی، این بخش از استاندارد ISO/IEC 15961 اکنون شامل پشتیبانی برای یک فرآیند «کاملا سراسری»[1] می‌باشد.

یادآوری- انتخاب مجزا بین گزینه‌هایی که کدبندی انتقالی و یک فرآیند «کاملا سراسری» را پشتیبانی می‌کنند اکنون واضح‌تر از اولین ویرایش استاندارد ISO/IEC 15961 تعریف می‌شود.

دو روش برای اجرای فرمان‌ها و پاسخ‌های تعریف شده در بند ۱۰ وجود دارد.

گزینه الف: فرآیند کاملا سراسری

در افزاره‌ها یا نرم‌افزارهایی که داده‌های ورودی فرمان‌ها را مطابق با این استاندارد با هم ترکیب می‌کنند و همچنین از فرآیندهای کدبندی استاندارد ISO/IEC 15962 پشتیبانی می‌کنند، کدبندی انتقالی لازم نمی‌باشد. بنابراین لازم است که مطمئن شویم کارکرد اصلی فرمان‌ها و پاسخ‌ها دنبال می‌شوند. قواعد تعریف شده برای Object Identifiers (به زیربند ۷-۳-۲ مراجعه کنید) و متغیرهای فرمان و پاسخ (به زیربند ۷-۴ مراجعه کنید) باید دنبال شوند.
شکل‌های مختلفی از پیاده‌سازی برای این فرآیند کاملا سراسری Data Protocol امکان‌پذیر هستند، از جمله ورودی از شکل‌ها و انتقال مستقیم‌تر از سامانه‌های میزبان. این استاندارد هیچ محدودیتی را برای این فرآیند انتخاب شده قرار نمی‌دهد.

گزینه ب: استفاده از کدبندی انتقالی

اگر یک افزاره یا نرم‌افزار، تنها فرآیندهای این استاندارد را پشتیبانی کند و یا تنها فرآیندهای کدبندی استاندارد ISO/IEC 15962 را پشتیبانی کند، سپس کدبندی انتقالی اصلی ممکن است استفاده شود (به پیوست الف مراجعه کنید). کدبندی انتقالی دیگری که شامل کدبندی دودویی است که در استاندارد ISO/IEC 24791-5 تعریف شده است ممکن است برای مجتمع‌سازی بهتر با سامانه‌ها استفاده شود تا از این سازوکارهای واسطه پشتیبانی کند. در عوض، این به معنای این است که ساختارهای فرمان و پاسخ تعریف شده در بند ۷ باید به عنوان چکیده یا تعریفات کارکردی الزامات مورد نیاز در نظر گرفته شوند تا داده‌ها بر روی یک برچسب خوانده و تفسیر شوند.

1 - Straight Through

۷-۲ سامانه کدبندی مربوط به اطلاعات در فرمان‌ها

۷-۲-۱ Singulation-Id

Singulation-Id (که از قبل به عنوان TagId دراستاندارد ISO/IEC 15961: 2004 تعریف شده است) با Tag Driver تهیه می‌شود و بدون ابهام، برچسب RFID را برای حداقل دوره تراکنش داده شناسایی می‌کند.

در پروتکل داده‌ها، **Singulation-Id** باید بیش از ۲۵۵ بایت باشد و همانند یک مرجع فایل برای Logical Memory عمل کند و در عوض یک اتصال مساوی با Logical Memory Map در خود برچسب RFID ایجاد کند. **Singulation-Id** باید براساس یکی از موارد زیر باشد، این انتخاب با طراحی Tag Driver و پروتکل واسط هوایی تعیین می‌شود:

الف) Unique Item Identifier به‌طورکامل برنامه‌ریزی شده در برچسب RFID، همان‌طور که در مجموعه استانداردهای ISO/IEC 18000 تعیین شده است. در آن مجموعه از استانداردها، با عنوان Tag ID تعریف می‌شود.

ب) یک شناسه مربوط به داده‌ها (مثال، همانند یک شناسه اقلام منحصر به فرد (UII)[1]، که برای منحصر به فرد بودن در قلمرو کاربردی خاص مدیریت اقلام می‌باشد. برای اثبات **Singulation-Id**، نیاز است که UII خوانده شود.

ج) یک شناسه مجازی یا نشست که مبتنی بر بازه زمانی یا دیگر ویژگی‌های مدیریت شده توسط پروتکل واسط هوایی است.

د) ترکیبی از موارد الف و ب. مثلا، یک **Singulation-Id** مجازی در واسطه هوایی، اما به شناسه داده‌ها نیاز دارد تا به عنوان یک پاسخ برگشت کند.

۷-۲-۲ AFI

AFI (که از قبل به عنوان ApplicationFamilyId در استاندارد ISO/IEC 15961: 2004 تعریف شده است) به شناسه‌های خاصی اشاره می‌کند که آدرس‌دهی انتخابی برچسب‌های FRID را فعال می‌کند.

این ممکن است با یک سازوکاری در واسط هوایی پشتیبانی شود. زیرا **AFI** با استانداردهایی برای کارت هوشمند پشتیبانی می‌شود. این ساختاری که دنبال می‌شود مخالف آن استانداردها نمی‌باشد. مقدار **AFI** برای RFID برای Item Management باید یک بایت باشد (در قسمت ۳ از استاندارد ISO/IEC 15961 تعیین شده است) و به شرح زیر می‌باشد:

- اگر در دامنه OO_{16} تا OF_{16} این کد در یک برنامه کاربردی سامانه بسته به کار برده شود، همان‌طور که در قسمت ۳ از استاندارد ISO/IEC 15961 تعریف شده است. این فهرست از مقادیر کدها، در انباره ساختارهای داده‌های مربوط به RFID به‌روزرسانی می‌شود.

1 - Unique Item Identifier

- اگر در دامنه 90_{16} تا CE_{16} این کد در یک برنامه کاربردی سامانه باز استفاده شود، همان‌طور که در قسمت ۳ از استاندارد ISO/IEC 15961 تعریف شده است. این فهرست از مقادیر کدها، در انباره ساختارهای داده‌های مربوط به RFID به‌روزرسانی می‌شود.
- مقدار CF_{16} به عنوان یک کد توسعه برای مقادیر بایت چندگانه کد **AFI**، کنار گذاشته می‌شود.

AFI باید در چند شکل بر روی برچسب RFID ذخیره شود و یا اگر اینها به‌طورکامل مخصوص باشند، ممکن است طور دیگری با خدمات واسط هوایی تعیین شود. اگر **AFI** با یک مجموعه برچسب RFID پشتیبانی نشود و خدمات از واسط هوایی این را با وسایل دیگری فراهم نکنند، مقدار این کد در اطلاعات سامانه برای برچسب RFID باید OO_{16} باشد.

یادآوری- در اولین ویرایش استاندارد ISO/IEC 15961 (و در فرمان‌ها و پاسخ‌های اولیه)، مقدار ApplicationFamilyId، هنگامی‌که در داخل اطلاعات سامانه بر روی برچسب RFID ذخیره می‌شوند، شامل دو مقدار صحیح است که در یک دسته هشتایی منفرد به هم پیوسته‌اند. اولین عدد صحیح به ApplicationFamily اشاره می‌کند و کدهای بعد از آن باید به صورت زیر به کاربرده شوند:

0: برای سامانه‌های بسته

۹ تا ۱۲: همان‌طور که در قسمت ۳ استاندارد ISO/IEC 15961 تعریف شده است.

دومین عدد صحیح ApplicationFamilyid برای ApplicationFamilies ، ۰ و ۹ تا ۱۲ می‌باشد که به ApplicationSubFamily اشاره می‌کند. مقادیر صحیح به یک مقداری معادل **AFI** با یک بایت تبدیل می‌شوند.

۷-۲-۳ DSFID

DSFID (که از قبل با عنوان Storage Format در استاندارد ISO/IEC 15961: 2004 تعریف شده است) به شناسه‌های ویژه اشاره می‌کند که کدبندی بهتری را بر روی برچسب‌های RFID فعال می‌کنند. مقدار **DSFID** برای RFID برای Item Management یک یا چند بایت است. **Access-Method** و **Data-Format** را کدبندی می‌کند که هر دو در قسمت زیر تعریف می‌شوند. **Access-Method** در بیت‌های ۸ و ۷ متعلق به **DSFID** کدبندی می‌شود و با پنج بیت نهایی (بیت‌های ۵ تا ۱) **Data-Format** را کدبندی می‌کند. بیت ۶ کارکرد اضافی را تعریف می‌کند که با چند نوع برچسب RFID پشتیبانی شده‌اند.

اگر **Access-Method** یک مقدار بین ۴ و ۱۵ داشته باشد، توسعه‌ها در **DSFID** توسط Data Processor انجام می‌گیرد.

یادآوری- هیچ‌کدام از این مقادیر تاکنون معین نشده‌اند و این تخصیص‌ها طبق تصحیح در این استاندارد می‌باشد.

اگر **Data-Format** یک مقداری بین ۳۲ و ۲۸۷ داشته باشد، همان‌طور که در قسمت ۳ استاندارد ISO/IEC 15961 تعریف شده است، توسعه‌ها در **DSFID** با Data Processor انجام می‌گیرند. زمانی که **Data-Format** به عنوان بخشی از استاندارد ISO/IEC 15961-2 Data Constructs ثبت می‌شود، این توسعه، مستقل از تغییرات استاندارد روی می‌دهد.

۷-۲-۴ Access-Method

Access-Method روشی را تعریف می‌کند که در آن داده‌ها را می‌توان بر روی برچسب RFID طراحی کرد و از طریق برچسب RFID می‌توان دسترسی داشت. مقدار **Access-Method** باید بر روی برچسب RFID ذخیره شود و یا ممکن است توسط خدمات واسط هوایی مشخص شود، البته اگر این را بتوان بدون ابهام انجام داد. **Access-Method** به عنوان یک مقدار بیت تعریف می‌شود و کدهای زیر باید به کار برده شوند:

۰ **No-Directory**- این **Access-Method** ساده‌ترین مجموعه از قواعد کدبندی را تهیه می‌کند. قانون اصلی، پیوند یک **Object** داده به یک **Object-Identifier** می‌باشد با مؤلفه‌های قواعد نحوی مناسب پشتیبانی می‌کند تا یک **Data-Sets** را ایجاد کند. **Data-Sets** به هم متصل می‌شوند تا یک ترتیب پیوسته‌ای از بایت‌های کد شده را در Logical Memory Map تولید کنند.

۱ **Directory**- ساختار **Directory** از تمام قواعد کدبندی ساختار **No-Directory** پشتیبانی می‌کند، اما همچنین از کدبندی یک فهرست راهنما در مقادیر آدرس بالاتر در Logical Memory Map بر روی برچسب RFID پشتیبانی می‌کند. زمانی که ساختار **Directory** استفاده می‌شود، هر دستوری که جستجو می‌کند تا بطور انتخابی داده‌ها را بخواند، می‌تواند دسترسی اولیه به فهرست داشته باشد تا آدرس حافظه را از شروع **Data-Set** پیدا کند، سپس به آن موقعیت می‌رود تا بایت‌هایی را بخواند که کدبندی **Data-Set** را نشان می‌دهد.

یادآوری- Directory ممکن است دیرتر از داده‌های کد شده اولیه در RFID نوشته شود.

۲ **Packed-Objects**- ساختار **Packed-Object** در اولین تجدیدنظر استاندارد ISO/IEC 15962 و این ویرایش این استاندارد معرفی شده است. طرح کدبندی متفاوت است. زیرا مجموعه‌ای از **Object-Identifiers** و **Object**های آن‌ها را می‌گیرد و آن‌ها را در یک ساختار اندیس‌گذاری شده، کدبندی می‌کند که فشرده‌سازی و کدبندی را یکپارچه می‌کند. طرح کدبندی **packed-Object** به یک جدول مبتنی بر قانون نیاز دارد که برای هر **Data-Format** لازم است که طرح‌هایی فشرده‌سازی ویژه برای هر یک از اقلام درخواست کند. طرح‌های فشرده‌سازی استاندارد هستند و توجهی به **Data-Format** ندارند. طرح **Packed-Object** به قواعد کدبندی و رمزگشایی پیچیده‌تری نسبت به **No-Directory** و **Directory Access-Methods** نیاز دارد، اما کارآیی زیادی از کدبندی را حتی بر روی ساختار اصلی **No-Directory** ارائه می‌دهد زمانی که چند **Object-Identifiers** باید کدبندی شوند. این

می‌تواند مقدار حافظه لازم بر روی برچسب RFID را کاهش دهد و می‌تواند اندازه پیام منتقل شده به رابط هوایی را در پاسخ به یک فرمان خواندن کاهش دهد. دامنه‌های کاربردی باید یک جدول از **Relative-OIDs** اندیس‌گذاری شده را برای این **Access-Method** تعریف کنند که اجرا می‌شود. همچنین به عنوان یک روش دیگر در **No-Directory** یا **Directory Access-Methods** بر روی یک برچسب ویژه RFID به کار برده می‌شود، اما برچسب‌ها با هر یک از **Access-Methods** ممکن است در همان برنامه کاربردی با هم ترکیب شوند.

۳ **Tag-Data-Profile**- طرح **Tag-Data-Profile** از ساختارهای پیام ثابت، پشتیبانی می‌کند، بطور نمونه چند **Object-Identifiers** و **Objects**های آن‌ها باید کدبندی شوند. ساختار پیام ثابت برای کاربردهایی مناسب است که معقولانه همگن هستند و همچنین یک شرط ثابت و پایداری برای همان **Object-Identifiers** دارند که به‌طوردقیق بر روی تمام برچسب‌های RFID در برنامه کاربردی کدبندی شده‌اند. هر **Tag-Data-Profile** نیاز به ثبت دارد.

۴ **Multiple-Records**- **Multiple-Records** فرآیندی را کدبندی می‌کند که روی یک ساختار در **No-Directory**، **Packed-Objects** و **Tag-Data-Profile Access-Methods** برای چند نمونه از اینها قرار می‌گیرد و حتی ترکیبی از این **Access-Methods**، در همان حافظه منطقی کدبندی می‌شود. این از طریق معرفی یک MR-header کدبندی شده و یک مقدمه برای هر رکورد به دست می‌آید. این، امکان کدبندی هر یک از رکوردها را فراهم می‌کند تا به‌طورکامل مطابق با **Access-Methods** ذاتی باشد. همچنین با این قاعده که فهرستی از همان اقلام داده‌ها را در یک رکورد پشتیبانی کند، **Object-Identifier** را در رکوردهای جدا کدبندی می‌کند. سه کلاس اصلی از برنامه کاربردی وجود دارد که با **Access-Methods** پشتیبانی شده است و اینها را می‌توان با هم ترکیب کرد. یک کلاس، برای شکل‌های مختلف داده‌ها و صاحبان داده‌ها این امکان را فراهم می‌کند که همان برچسب RFID را تسهیم کنند، اما تحت کنترل صاحب اصلی برچسب می‌باشد. کلاس دیگر یک ترتیب زمانی از رکوردهای تاریخی را از همان نوعی که تکرار می‌شود پشتیبانی می‌کند. سومین کلاس اصلی برای پشتیبانی از رکوردهای مربوط به سلسله مراتب است. مثلا، برای ارسال زنجیره ذخیره یا صورت‌حساب انواع مواد ساختاری می‌باشد.

۵ تا ۱۵ برای طرح‌های کدبندی آینده ذخیره می‌شود که در استاندارد ISO/IEC 15962 تعریف می‌شود و طبق قواعد برای یک **DSFID** با بایت چندگانه فراخوانی شود. بعضی از **Access-Methods** جدید ممکن است طرح‌های کدبندی ذاتی در **Multiple-Records** باشند. این،

تنها به عنوان طرح‌های کدبندی تعیین و ایجاد می‌شوند.

زمانی که انتخاب صورت می‌گیرد، راهنماهای زیر می‌توانند کمک کنند:

- ساختار **No-Directory** در برچسب‌های RFID با ظرفیت حافظه کم بسیار مناسب است، زیرا خود فهرست راهنما یک بالاسری است که باید کدبندی شود. همچنین در جایی مناسب است که **Object-Identifier** کمتری کدبندی می‌شوند، بنابراین یک تابع خواندن پیوسته، تمام کدبندی (یا حداقل تعداد کافی بایت‌های کدبندی شده) را ارسال می‌کند تا بایت‌ها بتوانند تجزیه شوند و **Object-Identifier** و **Object** مربوطه را پیدا کنند.
- ساختار **Directory** بطور بدیهی بسیار مناسب است اگر شرایط، متفاوت از شرایطی باشند که در ساختار **No-Directory** مناسب هستند. همچنین برای برنامه‌های کاربردیی بهتر است که خواندن، نوشتن و یا اصلاح انتخابی یک یا چند **Object-Identifier** را از میان بسیاری درخواست کنند. در این موقعیت، فرآیندهای خواندن زیاد، برای ارسال فهرست راهنما به Data processor ممکن است متعادل باشند و زمان خواندن برای **Object-Identifier** انتخابی کوتاهتر باشد.
- ساختار **Packed-Objects** به کدبندی و رمزگشایی پیچیده‌تری نیاز دارند، اما یک پیشرفت مهمی را در کارایی کدبندی بر روی ساختار اصلی **No-Directory** نشان می‌دهد. همچنین زمانی‌که شرایط کدبندی نسبت به ظرفیت حافظه برچسب‌های RFID در برنامه کاربردی به‌طورنسبی بسیار فشرده هستند، بهتر است که در برنامه‌های کاربردی استفاده شوند.
- طرح **Tag-Data-Profile** برای برنامه‌های کاربردی بسیار مناسب است که در آنجا قلمرو همگن است (مثلا، یک زنجیره تامین یکپارچه عمودی که تمام تامین‌کنندگان برای همه مشتریان شناخته شده هستند) و تمام گروه‌ها موافق هستند که هر **Object-Identifier** باید کدبندی شود و مطابق با یک ساختار و فرمت مشترک برای هر **Object** داده باشد.

همان‌طور که **Access-Methods** باید با برنامه کاربردی تعیین شود، سنجش انتقال‌های داده‌ها باید امکان‌پذیر باشد که یک ساختار داده‌های اضافی فهرست راهنما را با یک ساختار **No-Directory** شبیه‌سازی می‌کند. چنین آزمایشی را می‌توان برای تعیین نقطه شکست مناسب بین این و سایر **Access-Methods** استفاده کرد. تنوع و طول داده‌ها مورد توجه قرار می‌گیرند و نمونه پیاده‌سازی‌ها یا اجرای خواندن/ نوشتن برای یک برنامه کاربردی در نظر گرفته می‌شوند.

زمانی که یک **Access-Method** برای برچسب RFID تعیین شد، برنامه کاربردی نیازی نیست که تعیین کند که چگونه خواندن، نوشتن و یا سازمان‌دهی بایت‌ها بر روی Logical Memory به دست می‌آید. این، کارکرد Data Processor و Tag Driver است.

۷-۲-۵ Data-Format

Data-Format، **Root-OID** را از **Object-Identifier** تعریف می‌کند که بر روی برچسب RFID ذخیره می‌شود. **Data-Format**، **Object**های داده‌ها را فعال می‌کند تا توسط یک **Relative-OID**، به سادگی شناسه باشند، و از فضای کدبندی بهتر استفاده کنند و یا داده‌ها را در یک کلاس محدود کنند.

Object-Identifier کامل (یعنی با یک **Root-OID** متفاوت که با **Data-Format** تعریف می‌کنند) ممکن است همچنان بر روی برچسب RFID کدبندی شوند و داده‌ها را از قلمروهای کاربردی مختلف فعال کنند تا کدبندی شوند، اگرچه سطح کارآیی کدبندی همانند نمی‌باشد.

مقدار **Data-Format** باید در برچسب RFID ذخیره شود و یا ممکن است با خدمات واسط هوایی تعریف شود اگر این را بتوان بدون ابهام انجام داد. زمانی که با پروتکل واسط هوایی تعریف می‌شود، یک رابطه یک به یک با نوع برچسب RFID و مقدار **Data-Format** باید وجود داشته باشد. مقدار **Data-Format** برای RFID به منظور Item Management باید مقدار صحیح در دامنه 0 تا ۲۸۷ باشد که در قسمت ۳ استاندارد ISO/IEC 15961 تعیین شده است. موارد زیر به این بخش از استاندارد ISO/IEC 15961 مربوط هستند:

۰ **Not-Formatted**- این برای برچسب‌های RFID استفاده می‌شود که طبق این استاندار شکل‌دهی نشده است و یا هنوز شکل‌دهدی نشده است.

۱ **Full-Featured**- این **Data-Format** هر نوع شکل داده را پشتیبانی می‌کند که در آن‌ها **Object-Identifier** کامل استفاده می‌شود. هدف اولیه آن فعال کردن داده‌های ناهمگن می‌باشد (یعنی از فرهنگ لغات مختلف داده‌ها) که بر روی یک برچسب RFID کدبندی می‌شود. برای مثال این را می‌توان بدون ابهام، برای کدبندی داده‌های برنامه‌های کاربردی مختلف سامانه بسته استفاده کرد، با استفاده از **Object-Identifier** ثبت شده در استاندارد ISO/IEC 9834-1 برای هر برنامه کاربردی.

۲ **Root-Oid-Encoded**- این **Data-Format** زمانی استفاده می‌شود که تمام داده‌ها بر روی برچسب RFID، یک **Root-OID** مشترک دارند که مطابق با یکی از **Root-OIDs** معین مربوط به یک **Data-Format** که در قوانین قسمت ۲ استاندارد ISO/IEC 15961 ثبت شده است، نمی‌باشد.

Root-OID برای 2 **Data-Format** باید به‌طورمستقیم در برچسب RFID با استفاده از کمان‌های ریشه مناسب کدبندی شود. هر **Object** بر روی برچسب RFID با استفاده از یک **Relative-OID** کدبندی می‌شود که قوس‌های مرتبه

پایین‌تر باقیمانده را نشان می‌دهد.

۳ تا ۲۸	این مقادیر که در یک برنامه کاربردی سامانه باز به کاربرده می‌شود همان‌طور که در قسمت ۲ استاندارد ISO/IEC 15961 ثبت شده است. فهرست مقادیر رمز در ثبت ساختارهای داده‌های RFID مربوطه به‌روزرسانی می‌شود.
۲۹	این رمز در داده‌های سامانه بسته به کاربرده می‌شود که در آنجا داده‌های کدبندی شده موافق با قوانین این استاندارد و استاندارد ISO/IEC 15962 می‌باشد.
۳۰	این رمز در داده‌های سامانه بسته به کار برده می‌شود که در آنجا قوانین کدبندی موافق با ISO/IEC 15962 نمی‌باشند. در برنامه‌های کاربردی استفاده می‌شود که از قوانین کدبندی دیگر انتقال می‌یابند و داده‌ها را می‌توان تشخیص داد و به درستی کدبندی کرد. همچنین قبل از انتقال برچسب‌های RFID از یک برنامه سامانه بسته به یک برنامه کاربردی سامانه باز استفاده می‌شود.
۳۱	این مقدار به هیچ یک از برنامه‌های کاربردی تخصیص داده نمی‌شود زیرا دودویی معادل آن، برای علامت‌دهی به توسعه **Data-Format**، در دامنه ۳۲ تا ۲۸۷، به کار می‌رود.
۳۲ تا ۲۸۷	این مقادیر برای برنامه‌های کاربردی سامانه باز، کنار گذاشته شده اند (همان‌طور که در قسمت ۲ استاندارد ISO/IEC 15962 ثبت شده است) در صورتی‌که یک سازوکار توسعه‌یافته طبق قوانین استاندارد ISO/IEC 15962 برای یک **DSFID** بایت چندتایی، درخواست بشود.

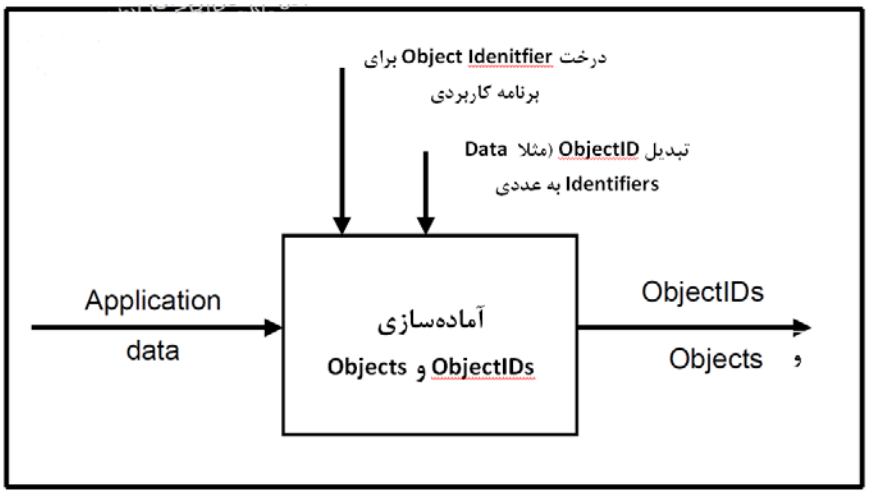
۷-۳ تهیه اشیاء اولیه و سایر متغیرهای مبتنی بر برنامه کاربردی

این یک فرآیند اولیه است تا مطمئن شویم که **Object**های داده‌ها در قالبی تهیه می‌شوند که مطابق با این استاندارد می‌باشند. تشخیص داده می‌شود که پروتکل‌های مبتنی بر پیغام و قواعد نحوی برای استانداردهای کاربردی AIDC موجود، وجود دارند که از پروتکل مبتنی بر شی این استاندارد متفاوت می‌باشد. می‌توان مزیت‌های پروتکل مبتنی بر شی را به دست آورد و سازگاری با سامانه‌های پیغام را با تبدیل بین دو قالب حفظ کرد (به پیوست ب مراجعه کنید).

خروجی‌های توصیف شده در زیر بندها که دنبال می‌شوند باید قالب ورودی در Data processor باشند. بنابراین سازوکارها برای به دست آوردن آن‌ها در ورودی و خروجی وجود ندارند. الزامات این است که یک **Object-Identifier** به هر **Object** با استفاده از فرهنگ لغات داده‌های مربوط به استاندارد برنامه کاربردی اختصاص یابد.

۷-۳-۱ مدل کلی

شکل ۵- مدل جریان داده‌ها: آماده‌سازی Objectهای اصلی- یک مدل داده را برای آماده‌سازی هر **Object-Identifier** و **Object** داده‌های مربوطه فراهم می‌کند. هر نوع داده کاربردی و به‌خصوص آن‌هایی که با این استاندارد پوشیده شده‌اند یک فرهنگ لغات داده‌ها را دارند و یا فهرستی از **Objects** داده‌ها یا اقلام داده‌ها را دارند.



شکل ۵- مدل جریان داده‌ها: آماده‌سازی Objectهای اصلی

هر یک از فرهنگ لغات داده ممکن است بیش از یک دوره زمانی ظاهر شود و نگهداری از این فرهنگ لغات داده باید جدا از این استاندارد باشد. فرهنگ لغات داده به‌طورمعمول شامل یک فهرست رمز شده (عددی، الفبایی، الفبا عددی و غیره) از **Objects** داده‌ها و مشخصات آن‌ها برای استفاده در حوزه استاندارد برنامه کاربردی می‌باشد. قسمت ۳ استاندارد ISO/IEC 15961 و ثبت ساختارهای داده‌های RFID مربوط به آن، **Object-Identifier** را تعریف می‌کند که به برنامه‌های کاربردی خاصی مربوط هستند.

۷-۳-۲ Object-Identifier

Object-Identifier باید با سامانه کاربردی به صورت یک سری قوس‌ها تهیه شود (همان‌طور که در بند ۲-۶ توصیف شده است) که مطابق با استاندارد ISO/IEC 9834-1 می‌باشد، شامل تمام **Object-Identifier** مربوط به **Data-Format** ایجاد شده. **Object-Identifier** باید به عنوان یک **Object-Identifier** (کامل) در دستورات برنامه کاربردی ارائه شود، مگر اینکه یک سامانه خاص- برنامه کاربردی تمام **Relative-OIDs** مربوط

را پشتیبانی کند. در هر زمان ممکن، Data processor یک **Object-Identifier** (کامل) را به یک **Relative-OIDs**برای کدبندی بهتر بر روی برچسب RFID تبدیل می‌کند که براساس **Data-Format** دریافت شده از برچسب RFID می‌باشد.

۷-۳-۳ Object-Identifiers مربوطه

قواعد نحوی پیام می‌توانند از روش‌های بازگشتی یا ایجاد حلقه استفاده کنند تا ترتیبات تکراری از داده‌های مربوطه را بسازند (مثال، کمیت فردی و اعداد دسته‌ای مربوط به کدهای محصولات مختلف).

زمانی که پیام کامل، تجزیه می‌شود، قواعد نحوی نقاط مرزی را شناسایی می‌کند بنابراین این مشخصه‌ها بطور صحیح به کد اولیه مربوط می‌شوند.

با یک سامانه مبتنی بر شی (از قبیل Data protocol از این استاندارد و استاندارد ISO/IEC 15962) که در یک سطح مبنا عمل می‌کنند، یک خطر ایجاد اتصال‌های غلط وجود دارد (یعنی کد محصول الف را می‌توان به کمیت محصول ب ارتباط داد). این مسئله با استفاده از روش‌هایی حل می‌شود که در پیوست ج توصیف شده است.

۷-۳-۴ Object

برنامه کاربردی داده‌ها را به شکل یک **Object** فراهم می‌کند. این مقدار براساس بایت می‌باشد و به تعریف **Object-Identifier** مربوط می‌شود که با فرهنگ لغت داده‌های کاربردی تهیه شده است. توصیه مخصوص در رابطه با ترجمه یا تفسیر خاص در زیر بندهای زیر ارائه می‌شود.

۷-۳-۴-۱ اطلاعات مجموعه نویسه ۸ بیتی اصلی

بسیاری از کاربردهای تجاری، از استاندارد ISO/IEC 8859-1 یا زیر مجموعه آن استاندارد ISO/IEC 646 استفاده می‌کنند. همین‌طور، تفسیر (یا ارائه) داده‌ها برای تمام شریک‌ها در سامانه معلوم است. این شامل فرستنده مخصوص و حتی دریافت کننده مخصوص می‌باشد، همین‌طور در یک سیتسم AIDC باز مشترک می‌باشد، آن‌ها به‌طورمستقیم برای یکدیگر معلوم نیستند، اگر فایل متنی و نرم‌افزار کاربردی مرورگر تنظیم شوند تا مجموعه نویسه استاندارد ISO/IEC 8859-1 را کنترل کنند، بیشترین حجم داده‌ها را می‌توان نمایش داد. اگر این حجم یک تفسیر خاصی دارد، برنامه کاربردی RFID باید مشخص کند که بین شریک‌ها به‌طوردقیق چگونه داده‌های ویژه تفسیر می‌شود. این را می‌توان با انتشار یک فرهنگ لغت داده‌ها با استفاده از **Object-Identifier** به عنوان یک مرجع به دست آورد.

۷-۳-۴-۲ پشتیبانی برای استاندارد ISO/IEC 10646

استاندارد ISO/IEC 10646 از علامت‌های نویسه[1] تمام مجموعه‌های نویسه پشتیبانی می‌کند. یک گزینه برای پشتیبانی داده‌ها در هر مجموعه از نویسه ویژه، پیش‌پردازش آن با تبدیل استاندارد ISO/IEC 10646 با استفاده از ابزارهای نگاشت می‌باشد و سپس آن را با کدبندی UTF-8،فشرده می‌کنند (همان‌طور که در استاندارد

1 - Character glyphs

ISO/IEC 10646 تعریف شده است). هر داده‌ای که به‌طورمستقیم با استاندارد ISO/IEC 10646 تعریف شده است باید با UTF-8 کدبندی شده باشد، تا تعداد بایت‌های لازم را کاهش دهند و **Object** در Logical Memory Map ذخیره شود.

۷-۳-۴-۳ پشتیبانی برای داده‌های مطمئن یا رمزنگاری شده

همه شکل‌های داده‌های مطمئن، از جمله داده‌های رمزنگاری شده، باید قبل از اینکه بعنوان یک Object برای Data Processor عبورداده شوند[1]، باید بوسیله برنامه کاربردی ساخته شوند. این به دو دلیل انجام می‌گیرد:

- نیاز است تا امنیت داده درون برنامه کاربردی ساکن شود و بطور مستقل از Data Processor، تغییر کند.
- به Data processor اجازه دهد تا تمام داده‌ها را مدیریت کند، چه رمزنگاری شده باشند و یا نشده باشند و وضعیت آن‌ها همانند است.

واقعیت این است که داده‌هایی که رمزنگاری می‌شوند ممکن است در تمام مجموعه کاربران سامانه‌ها شناخته شده باشند، اما روش واقعی رمزنگاری را می‌توان به دریافت کننده مورد انتظار و فرستنده محدود کرد.
جزئیات بیشتر در زیر بند ۷-۶ فراهم می‌شوند.

۷-۳-۵ Compact-Parameter

شمول **Compact-Parameter** در یک فرمان باید تعیین کند که آیا داده‌های برنامه کاربردی باید با Data processor فشرده شوند. **Compact-Parameter** یک مقدار صحیح است و کدهای زیر به کاربرده می‌شوند.

۰ **Application-Defined Object**- نباید از طریق قوانین فشرده‌سازی داده‌های استاندارد ISO/IEC 15962 پردازش شود و بدون تغییر می‌ماند زمانی که در Logical Memory Map برچسب RFID ذخیره می‌شود.

۱ **Compact**- استفاده از قوانین فشردگی اصلی استاندارد ISO/IEC 15962 برای فشرده کردن Object لازم است تا بتواند تعداد بایت‌های لازم بر روی Logical Memory Map را کاهش داد.

۲ **UTF8-Data**- این شناسایی می‌کندکه **Object** در خارج از مجموعه نویسه رمز شده استاندارد ISO/IEC 10464 به کدبندی UTF-8 تغییرشکل داده است. **Object** نباید از طریق قوانین فشرده‌سازی داده‌های استاندارد ISO/IEC 15962 پردازش شود و برای انتقال به Logical Memory Map بدون تغییر می‌ماند.

1 - Pass

۳ **Pack-Objects**- این مشخص می‌کند که یک مجموعه‌ای از **Object** با استفاده از طرح کدبندی Packed Object کدبندی می‌شود و با **Access-Method** مربوطه شناسایی می‌شود. مجموعه **Objects** نباید با استفاده از قوانین اصلی فشرده‌سازی استاندارد ISO/IEC 15962 فشرده شود، اما از قوانین کدبندی Packed Object استفاده می‌کند.

۴ Tag-Data-Profile- این مشخص می‌کند که یک مجموعه‌ای از **Objects** با استفاده از طرح کدبندی Tag Data Profile کدبندی می‌شود و با **Access-Method** مربوط مشخص می‌شود. اگرچه مجموع طرح‌های فشرده‌سازی در قوانین اصلی فشرده‌سازی استاندارد ISO/IEC 15962 یکسان هستند، Profile ID Table یک طرح فشرده‌سازی خاصی را تعیین می‌کند. مشخصات Profile ID Table باید دنبال شود.

۵ **Monomorphic-UII**- این مشخص می‌کند که یک **Object** واحد که یک UII را تعریف می‌کند با استفاده از قوانین فشرده‌سازی تعریف شده با **AFI** کدبندی می‌شود. همان‌طور که این در یک UII به کار برده می‌شود، **Compact-Parameter** باید تنها در یک **Object** واحد در هر بانک حافظه به کار برده شود.

یادآوری- این **Compact-Parameter** به‌طورصریح تعریف می‌شود زیرا برای بعضی برچسب‌ها کدبندی Objects جدا از کدبندی AFI می‌باشد، بنابراین Data Processor به اطلاعاتی درمورد **Compact-Parameter** از ثبت ساختارهای Data دسترسی دارد همان‌طور که در قسمت ۲ استاندارد ISO/IEC 15961 تعریف شده است.

۶ تا ۱۳ برای تعریف آینده ذخیره شده است.

۱۴ **De-Compacted-Monomorphic-UII**- این مشخص می‌کند که **Object** در یک پاسخ با استفاده از قوانین تعریف شده توسط **AFI**، فشرده شده است که برای یک **Monomorphic-UII** ویژه اختصاص یافته است.

یادآوری- این **Compact-Parameter** به‌طورصریح تعریف می‌شود زیرا **Monomorphic-UII**، یک **Object-Identifier** مربوطه که بر روی برچسب کدبندی شده است، را ندارد. **Object-Identifier** همان‌طور که توسط AFI تعریف شد، باید به این پاسخ اضافه شود. Data Processor باید به اطلاعاتی درمورد **Object-Identifier** از انباره Data constructs دسترسی داشته باشد همان‌طور که در قسمت ۲ استاندارد ISO/IEC 15961 تعریف شده است.

۱۵ **De-Compacted-Data**- این مشخص می‌کند که **Object** در یک پاسخ، با استفاده از قوانین در استاندارد ISO/IEC 15962 دوباره فشرده شده است و در قالب ورودی برنامه کاربردی اصلی آن، برگردانده شده است.

بطور کلی، فشرده‌سازی باید به کار برده شود زیرا کارآیی کدبندی را بر روی برچسب RFID افزایش می‌دهد. **Object**هایی که از قبل با قوانین کدبندی UTF-8 کدبندی شده‌اند باید با مقدار **Compact-Parameter** (۲) تعیین شوند بنابراین خواندن بعدی **Object**، بطور واضح نشان می‌دهدکه این باید از طریق یک رمزگشای UTF-8 برای ارائه نهایی به برنامه کاربردی پردازش شود که داده‌ها را دریافت می‌کند. مقدار **Compact-Parameter** (۰) تنها باید زمانی استفاده شود که یک متغیر جایگزین برای فشرده نشدن داده‌ها وجود داشته باشد. متغیر برای این شامل فشرده‌سازی قبلی در قوانین برنامه کاربردی می‌باشد و یا اینکه آیا **Object** رمزنگاری شده است. در هر دو مورد، اگرچه فشرده‌سازی ممکن است امکان‌پذیر باشد و فرآیند رمزگشایی، **Object** را بطور کامل برگرداند، مقدار پارامتر اصلی تعریف شده ۰ یا ۲ از بین می‌رود و برنامه کاربردی دریافت شده پردازش بعدی را تقبل نمی‌کند.

یک روش دیگر برای فشرده‌سازی اصلی، برای مجموعه‌ای از **Objects** می‌باشد که با قوانین Packed Objects کدبندی می‌شود. این را می‌توان انجام داد اگر مدیران برنامه کاربردی انتخاب کنند که طرح را بوسیله ساخت یک جدول اندیس بپذیرند که درآن مرجع‌های منبع باید در انباره ساختارهای داده تهیه شوند. سپس برای هر برچسب RFID که باید با قوانین **Packed Objects** کدبندی شود، مقدار **Compact-Parameter** (۳) در مجموعه کامل **Objects** به کار برده می‌شود تا کدبندی شود.

همین‌طور، اگر داده‌ها طبق قوانین Tag Data profiles کدبندی شوند، مقدار **Compact-Parameter** (۴) در مجموعه کامل **Objects** به کار برده می‌شود تا کدبندی شود.

در طی فرآیند رمزگشایی، کدهای پاسخ زیر براساس شرایط ورودی به کار برده می‌شوند:

- اگر متغیر فرمان، **Application-Defined** (۰) باشد، یک Data Processor موافق، هیچ فشرده‌سازی را تقبل نمی‌کند و از رمز فشرده مناسب برای کدبندی در برچسب استفاده می‌کند. همان‌طور که رمزگشا نمی‌تواند بایت‌های کدبندی شده را تفسیر کند، پاسخ، **Application-Defined** (۰) می‌باشد.
- اگر متغیر فرمان، **Compact** (۱) باشد، Data Processor موافق، برای کدبندی در برچسب از کد فشرده صحیح استفاده کرده است. همان‌طور که رمزگشا می‌تواند بایت‌های کدبندی شده را تفسیر کند، پاسخ، **De-Compacted-Data** می‌باشد (۱۵).
- اگر متغیر فرمان، **UTF8-Data** (۲) باشد، یک Data Processor موافق، هیچ فشرده‌سازی را تقبل نکرده است و از کد فشرده مناسب برای کدبندی در برچسب استفاده کرده است، همان‌طور که رمزگشا نمی‌تواند بایت‌های کدبندی شده را تفسیر کند، پاسخ، **UTF8-Data** (۲) می‌باشد.

- اگر متغیر فرمان، **Pack-Objects** (۳) باشد، یک Data Processor موافق، به قواعدی کدبندی می‌کند که مانند قواعد تعریف شده برای **Object** ویژه است و همانند ID Table مشخص شده است. رمزگشا از همان ID Table برای تفسیر بایت کدبندی شده استفاده می‌کند و هر Object داده را با پاسخ De-Compacted-Data (۱۵) بر می‌گرداند.

- متغیر فرمان، Tag-Data-Profile باشد (۴)، یک Data Processor موافق، به قواعدی کدبندی می‌کند که مانند قواعد تعریف شده برای **Object** ویژه است و همانند ID Table مشخص شده است. رمزگشا از همان ID table استفاده می‌کند تا بایت کدبندی شده را تفسیر کند و هر **Object** داده را با پاسخ **De-Compacted-Data** برگشت دهد.

- اگر متغیر فرمان، Monomorphic-UII (۵) باشد یک Data Processor موافق، به قواعدی کدبندی می‌کند که مانند قواعد تعریف شده برای **Object** ویژه است و همانند تعریف با **AFI** و ثبت ساختارهای Data می‌باشد. رمزگشا، با شناسایی AFI، از قانون فشرده‌سازی استفاده می‌کند که در انباره Data Constructs اعلان شده است و Object داده‌ها را با پاسخ De-Compacted-Monomorphic UII بر می‌گرداند (۱۴). همچنین Object-Identifier را بر می‌گرداند همان‌طور که با انباره Data Constructs تعریف شده است.

یادآوری- اگرچه رمزگشا باید قوانین فشرده‌سازی و فشرده‌سازی مجدد را جستجو کند، همان‌طور که با AFI از انباره Data Constructs اعلان شده است، پاسخ‌ها از داده‌های خواندن شامل AFI نمی‌باشند. این متغیر پاسخ لازم است تا مطمئن شویم که دستورالعمل‌های انطباق زیادی می‌توانند اجرا شوند.

۷-۳-۶ Object-Lock

متغیر فرمان **Object-lock**، به Data processor نیاز دارد تا مجموعه داده‌های کدبندی شده را مرتب کند (**Object-Identifier** یا **Relative-OID**، **Object**، Precursor و مؤلفه‌های قواعد نحوی مربوطه دیگر) بطوری‌که تمام بایت‌های مربوطه را می‌توان ذخیره کرد و در یک روش بلوکی در Logical Memory Map قفل کرد. فرآیند قفل شدن باید بر روی ساخت بایت‌ها اثر داشته باشد. بنابراین بطور ثابت پردازش شده‌اند و بر روی برچسب RFID کدبندی شده‌اند و یا تنها می‌توانند در چند نوع برچسب RFID با استفاده از کلمات عبور مناسب باز شوند.

۷-۴ سایر متغیرهای فرمان

۷-۴-۱ Access-Password

متغیر فرمان **Access-Password** نیاز به Data Processor دارد تا این را به بازپرس انتقال دهد بنابراین **Access-Password** در این فرمان مطابق با برچسب RFID می‌باشد. یک انطباق منجر به عمل‌های اضافی بر روی برچسب RFID می‌شود که مجاز می‌باشد. یک عدم انطباق موجب می‌شود تا واسط هوایی فرمان را رد کند.

۷-۴-۲ Additional-App-Bits

متغیر فرمان **Additional-App-Bits** استفاده می‌شود تا معیار جستجوی داده‌های **Object-Identifier** که در حافظه UII یک برچسب RFID حافظه قطعه‌بندی شده کدبندی شده است، تعمیم پیدا کند. **Additional-App-Bits** به عنوان یک پسوند به **AFI** و **DSFID** اضافه می‌شود.

۷-۴-۳ AFI-Lock

متغیر فرمان **AFI-Lock** برای این استفاده می‌شود تا تعیین کند که آیا **AFI** قفل می‌شود یا نه. این یک متغیر BOOLEAN است. اگر به TRUE تنظیم شود، تحقیق‌کننده باید **AFI** را قفل کند تا مطمئن شود که برچسب RFID ویژه تنها به روش مجاز استفاده می‌شود. فرآیند قفل شدن باید تاثیر ساخت بایت‌ها را داشته باشد بنابراین بطور ثابت پردازش شده و بر روی برچسب RFID کدبندی می‌شوند و یا تنها می‌توانند در چند نوع برچسب RFID با استفاده از کلمات عبور مناسب باز شوند.

یادآوری- در چند نوع برچسب، AFI بخشی از یک رشته پیوسته با مجموعه داده‌های کد شده می‌باشد. در چنین مواردی نمی‌تواند بطور مستقل قفل شود.

۷-۴-۴ Append-To-Existing-Multiple-Record

متغیر فرمان **Append-To-Existing-Multiple-Record** یک متغیر BOOLEAN است. اگر TRUE باشد، اشیاء داده‌ها به یک رکورد چندگانه موجود ضمیمه می‌شود که با Data processor بررسی‌های مختلفی می‌شود. اگر FALSE باشد، آنگاه یک رکورد چندگانه جدید باید ساخته شود.

۷-۴-۵ Application-Defined-Record-Capacity

متغیر فرمان **Application-Defined-Record-Capacity** یک متغیر BOOLEAN است که اگر FALSE باشد تعیین می‌کند که Data processor بطور خودکار تعیین می‌کند که ظرفیت برای رکورد چندگانه معین به سادگی براساس داده‌های رمز شده می‌باشد و سپس افزایش می‌یابد تا مطابق با بلوک باشد. اگر TRUE باشد، سپس برنامه کاربردی باید با استفاده از متغیر **Record-Memory-Capacity**، اندازه حافظه تخصیص یافته به رکورد را تنظیم کند.

۷-۴-۶ Avoid-Duplicate

متغیر فرمان **Avoid-Duplicate** اطمینان می‌دهد که **Object-Identifier** در Logical Memory Map کدبندی نمی‌شود. این یک متغیر BOOLEAN است. اگر به TRUE تنظیم شود، تحقیق‌کننده باید تمام **Object-Identifiers** را در Logical Memory Map بررسی کند. اگر **Object-Identifiers** وجود داشته باشد، فرمان نوشتن صرف‌نظر می‌شود و مقدار **Completion-code** از **Duplicate-Object** (۱۰) بر می‌گردد. اگر به FALSE تنظیم شود، تحقیق‌کننده باید Object را بدون هیچ وارسی بنویسید.

۷-۴-۷ Battery-Assist-Indicator

متغیر فرمان **Battery-Assist-Indicator** با برنامه‌ای استفاده می‌شود که برچسب RFID یک ویژگی کمکی باطری را پشتیبانی می‌کند. این متغیر بطور کلی زمانی استفاده می‌شود که پروتکل واسط هوایی ویژگیی را ندارد تا چنین نشانه‌ای را پشتیبانی کند.

۸-۴-۷ Block-Align

متغیر فرمان **Block-Align** استفاده می‌شود تا تعریف کنند آیا **Objects** نوشته شده در یک برچسب باید در شروع یک بلوک ردیف شوند. این یک متغیر BOOLEAN می‌باشد. اگر به TRUE تنظیم شود، تحقیق‌کننده باید هر **Object** نوشته شده در یک برچسب را بر روی یک مرز بلوک ردیف کند همان‌طور که توسط معماری یا ساخت برچسب ویژه تعیین شده است.

۹-۴-۷ Block-Align-Packed-Object

متغیر فرمان **Block-Align-Packed-Object** استفاده می‌شود تا تعیین کنند که آیا **Packed-Objects** نوشته شده در یک برچسب باید در شروع بلوک ردیف شوند. این یک متغیر BOOLEAN است. اگر به TRUE تنظیم شود، تحقیق‌کننده باید هر **Packed-Objects** نوشته شده در یک برچسب را روی یک مرز بلوک ردیف کند همان‌طور که با ساخت یا معماری ویژه تعیین شده است.

۱۰-۴-۷ Check-Duplicate

متغیر فرمان **Check-Duplicate** استفاده می‌شود تا یک فرآیند ویژه تعریف شده با فرمان را درخواست کنند (مثال خواندن یا حذف). این یک متغیر BOOLEAN است. اگر به TRUE تنظیم شود، تحقیق‌کننده باید تمام **Object-Identifiers** را در Logical Memory Map بررسی کند. این فرمان باید تنها زمانی کامل شود که هیچ نسخه کپی از آن وجود نداشته باشد. در غیر این‌صورت مقدار **Completion-Code** از **Duplicate-Object** (۱۰) بر می‌گردد.

اگر پرچم **Check-Duplicate** به FALSE تنظیم شود، تحقیق‌کننده باید بر روی اولین رویداد **Object-Identifier**، که یک **Object** مربوط و پیشرو است، عمل کند. در این حالت، یک **Object-Identifier** دو نسخه‌ای و **Object** مربوطه (ممکن است با یک مقدار مختلف) و پیش‌رو ممکن است وجود داشته باشد.

۱۱-۴-۷ Data-CRC-Indicator

متغیر فرمان **Data-CRC-Indicator** در یک فرمان Data Processor دستور می‌دهد تا یک CRC-16 را به هر مجموعه از داده‌ها و یا تمام داده‌های رمز شده اضافه کنند.

متغیر **Data-CRC-Indicator** در یک پاسخ نشان می‌دهد که Data Processor ، صحت CRC-16 را تعیین می‌کند و این را از پاسخ حذف کرده و **Object-Identifier** تفسیر شده را انتقال داده و **Object** مربوط به برنامه کاربردی را نیز انتقال داده است.

۷-۴-۱۲ Data-Length-Indicator

این متغیر پاسخ، اندازه Multiple Record کدبندی شده را برحسب اندازه بلوک نوشتن فراهم می‌کند، همان‌طور که در قالب EBV-8 در مقدمه رکورد کدبندی شده است.

۷-۴-۱۳ Delete-MR-Method

متغیر فرمان **Delete-MR-Method** توسط برنامه کاربردی استفاده می‌شود که یکی از دو روش را برای تحقیق‌کننده درخواست می‌کنند تا یک رکورد چندگانه را حذف کند. این متغیر فرمان به عنوان یک مقدار صحیح ارائه می‌شود و کدهای زیر به کاربرده می‌شوند:

- ۰ رکورد را بعنوان رکوردحذف شده علامت‌گذاری می‌کند.
- ۱ بطور فیزیکی رکورد را حذف می‌کند.

جزئیات فرآیند در زیربند ۱۰-۲۷-۱ تعریف می‌شوند.

۷-۴-۱۴ Directory-Length-EBV8-lndicator

متغیر فرمان **Directory-Length-EBV8-lndicator** با برنامه کاربردی استفاده می‌شود تا اطمینان دهد که فضای کافی در MR-header برای Data processor در دسترس می‌باشد تا طول واقعی فهرست راهنما را کدبندی کنند. این متغیر فرمان به صورت یک مقدار صحیح نشان داده می‌شود و کدهای زیر به کار برده می‌شوند:

۱ EBV-8 تک بایتی که می‌تواند از یک فهرست تا ۱۲۷ بلوک پشتیبانی کند.

۲ EBV-8 دو بایتی که باید زمانی استفاده شود که Logical Memory بزرگ است و/یا بسیاری از رکوردها بر روی آن کدبندی می‌شوند. این می‌تواند از یک اندازه فهرست راهنمای بالاتر از ۱۶۳۸۳ بلوک پشتیبانی کند.

مقدار ۲ برای یک فهرست راهنما استفاده می‌شود که ممکن است در اندازه کوچک باشد، اما در آینده ممکن است بزرگ‌تر شود.

۷-۴-۱۵ DSFID-Lock

متغیر فرمان **DSFID-Lock** (به‌طور سابق StorageFormatLock) استفاده می‌شود تا تعیین کنیم که آیا **DSFID** قفل می‌شود یا نه. این یک متغیر BOOLEAN است. اگر به TRUE تنظیم شود، تحقیق‌کننده باید DSFID را قفل کند تا مطمئن شود که برچسب ویژه RFID را می‌توان تنها به روش مجاز استفاده کرد. فرآیند قفل شدن باید بر ساخت بایت‌ها اثر داشته باشد بنابراین بطور ثابت با کدبندی بر روی برچسب RFID پردازش می‌شود و یا تنها ممکن است در چند نوع برچسب RFID با استفاده از کلمات عبور مناسب باز شود.

یادآوری- در بعضی انواع برچسب، **DSFID** بخشی از یک رشته پیوسته با مجموعه داده‌های رمز شده می‌باشد. در چنین مواردی نمی‌تواند جداگانه قفل شود.

DSFID-Pad-Bytes ۱۶-۴-۷

متغیر فرمان **DSFID-Pad-Bytes** تعداد بایت‌هایی را تعیین می‌کند که برنامه کاربردی می‌خواهد در یک **Extended-DSFID** تهیه کند علاوه بر آن‌هایی که با Data processor ساخته می‌شوند. برای مثال، این می‌تواند فضای کدبندی را برای طول داده‌های کد شده فراهم کند تا در آینده اضافه و به‌روزرسانی شود. هنگامی که در یک پاسخ قرار گرفته‌اند، بایت‌های خالی اضافی ممکن است توسط Data Processor اضافه شوند تا در یک مرز بلوک- قفل ردیف شوند اگر **Extended-DSFID** قفل می‌باشد و یا اگر اولین **Data-Set** قفل می‌شود.

Editable-Pointer-Size ۱۷-۴-۷

متغیر فرمان **Editable-Pointer-Size** استفاده می‌شود تا تعیین کنید که یک Packed Object که به‌طور صریح یا ضمنی از طریق یک فرمان **Write-Objects** ساخته شده است باید امکان تغییرات بعدی را در محتویات یا ساختار آن فراهم کند. این متغیر فرمان به عنوان یک مقدار صحیح غیرصفر نشان داده می‌شود که نشان می‌دهد Packed Object ساخته شده باید با اضافه کردن یک بخش فرعی ضمیمه اختیاری به پایان بخش Object Info از Packed Object، قابل ویرایش باشد. اشاره‌گرها در Addendum Packed-Object باید تعداد بیت‌های تعیین شده در این متغیر فرمان باشند. مقدار پیش فرض برای این متغیر صفر است که نشان می‌دهد هیچ بخش فرعی ضمیمه وجود ندارد و بنابراین Packed Object قابل ویرایش نمی‌باشد.

Encoded-Memory-Capacity ۱۸-۴-۷

این متغیر پاسخ، اندازه حفظ شده برای یک Multiple Record را برحسب اندازه بلوک نوشتن فراهم می‌کند، همان‌طور که در فرمت EBV-8 در مقدمه رکورد کدبندی شده است.

EPC-Code ۱۹-۴-۷

متغیر فرمان **EPC-Code** هر یک از کدهای انحصاری تعریف شده با استاندارد داده برچسب EPCglobal[1] را نشان می‌دهد. ساختار هر **EPC-Code** از مقدار سرآیند ۸ بیتی، خود- اعلان است. اگرچه چند کد در یک مرز ۸ بیتی ردیف نمی‌شوند، اینها را باید در بایت‌های ۸ بیتی برای پردازش در Data Processor گرد کرد[2] و در کلمات ۱۶ بیتی برای کدبندی در برچسب RFID مربوطه گرد کرد.

Full-Function-Sensor-indicator ۲۰-۴-۷

متغیر فرمان **Full-Function-Sensor-indicator** با برنامه کاربردی استفاده می‌شود که برچسب RFID یک حسگر با کارکرد کامل را پشتیبانی می‌کند. این متغیر بطور کلی زمانی استفاده می‌شود که پروتکل واسط هوایی ویژگیی که چنین نشانه یا علامتی را پشتیبانی کند، ندارد.

1 - EPCglobal Tag Data Standard
2 - Round

Hierarchical-Identifier-Arc ۲۱-۴-۷

متغیر پاسخ **Hierarchical-Identifier-Arc** یک مقدار صحیح است که از مقدار EBV-8 کدبندی شده در مقدمه Multiple Records از یک رکورد سلسله مراتبی، برگردانده شده است.

Identifier-Of-My-Parent ۲۲-۴-۷

متغیر فرمان و پاسخ **Identifier-Of-My-Parent** برای یک رکورد چندگانه استفاده می‌شود که بخشی از یک سلسله مراتب است و مقدار کد سلسله مراتبی را از آن رکورد دارد.

Identify-Method ۲۳-۴-۷

متغیر فرمان **Identify-Method** استفاده می‌شود که آیا همه یا برخی از برچسب‌های RFID، به **AFI** انتخابی خاص در زمینه عملیاتی که باید شناسایی شوند، تعلق دارد. این متغیر فرمان به صورت یک مقدار صحیح ارائه می‌شود و کدهای زیر به کار برده می‌شوند:

۰	**Inventory-All-Tags**
۱	**Inventory-At-Least**
۲	**Inventory-No-More-Than**
۳	**Inventory-Exactly**
۴ تا ۱۵	برای تعریف آینده کنار گذاشته شده است.

برای هر متغیر از جمله **Inventory-All-Tags**، لازم است که تعداد برچسب‌های RFID را مشخص کنیم تا با استفاده از **Number-Of-Tags** خوانده شوند (به زیربند ۷-۴-۴۳ مراجعه کنید)، توصیه دقیق‌تر در دستور **Inventory-Tags** فراهم می‌شود (به زیربند ۳-۱۰ مراجعه کنید).

ID-Type ۲۴-۴-۷

متغیر فرمان **ID-Type** تعیین می‌کند که یک **Packed-Object** ساخته شده بطور صریح یا ضمنی از طریق یک فرمان **Write-Objects** باید همانند نوع ID List یا ID Map معرفی شود. این متغیر فرمان به صورت یک مقدار صحیح نشان داده می‌شود و کدهای زیر استفاده می‌شوند:

۰	**ID List**
۱	**ID Map**
۲ تا ۱۵	برای تعریف آینده کنار گذاشته شده است.

Instance-Of-Arc ۲۵-۴-۷

متغیر پاسخ **Instance-Of-Arc** یک مقدار صحیح است که از مقدار EBC-8 کد شده در مقدمه Multiple Records برگردانده شده است.

Kill-Password ۲۶-۴-۷

متغیر فرمان **Kill-Password** به Data Processor نیاز دارد تا این را به تحقیق‌کننده انتقال دهد بطوریکه **Kill-Password** در این دستور مطابق با برچسب RFID می‌باشد. یک انطباق منجر به غیرفعال شدن دائمی کارکرد برچسب RFID می‌شود. یک عدم انطباق موجب می‌شود که واسط هوایی، فرمان را رد کند.

Length-Of-Mask ۲۷-۴-۷

متغیر فرمان **Length-Of-Mask** در اتصال با متغیرهای فرمان **Pointer** و **Tag-Mask** استفاده می‌شود تا معیار جستجوی داده‌های EPCglobal کد شده در حافظه UII را از یک برچسب RFID حافظه قطعه‌بندی شده تعریف کنند.

Lock-Directory-Entry ۲۸-۴-۷

متغیر فرمان **Lock-Directory-Entry**، BOOLEAN می‌باشد و اگر به TRUE تنظیم شود، تحقیق‌کننده باید ورودی فهرست راهنما را برای Multiple Record ویژه قفل کند.

Lock-Multiple-Records-Header ۲۹-۴-۷

متغیر فرمان **Lock-Multiple-Records-Header** استفاده می‌شود تا تعیین کنند که آیا تمام، بعضی یا هیچ‌کدام از MR-header قفل است. این متغیر فرمان به صورت یک مقدار صحیح ارائه می‌شود و از کدهای زیر استفاده می‌شود:

۰	قفل نشده
۱	بطور کامل قفل شده
۲	تعداد
۳	Data Length از فیلد Directory قفل نشده می‌ماند و سایر فیلدهای شامل فیلد Number of Records قفل می‌شود.
۴ تا ۱۵	Data Length از فیلد Directory و فیلد Number of Records، قفل نشده می‌ماند و سایر فیلدهای قفل می‌شود.

Lock-Record-Preamble ۳۰-۴-۷

متغیر فرمان **Lock-Record-Preamble**، BOOLEAN است اگر به TRUE تنظیم شود، تحقیق‌کننده باید رشته‌ها را در مقدمه قفل کند.

Lock-UII-Segment-Arguments ۳۱-۷-۴

متغیر فرمان **Lock-UII-Segment-Arguments** استفاده می‌شود تا تعیین کند کدام بخش‌های مؤلفه از قطعه UII از استاندارد ISO/IEC 18000-6 برچسب نوع D، باید قفل شوند. این متغیر نسبت به متغیرهای دیگر قفل، در فرمان مربوطه اولویت دارد.

Max-App-Length ۳۲-۴-۷

متغیر فرمان **Max-App-Length** برای تعریف حداکثر طول فشرده یک مجموعه داده‌های رمز شده استفاده می‌شود. در دستورهایی استفاده می‌شود که باید فرمان‌های خواندن واسط هوایی را برای تراکنش‌های سریع‌تر محدود کنند، مثلا، برای خواندن UII رمز شده در اولین مجموعه داده بر روی برچسب RFID.

Memory-Bank ۳۳-۴-۷

متغیر فرمان **Memory-Bank** استفاده می‌شود تا تعریف کند کدام بخش از برچسب حافظه قطعه‌بندی شده بر روی کدام داده باید کدبندی شود. سپس Data processor قوانینی را درخواست می‌کند که مخصوص **Memory-Bank** هستند تا یک برچسب رمز شده صحیح را به دست آورند.

Memory-Bank-Lock ۳۴-۴-۷

متغیر فرمان **Memory-Bank-Lock** استفاده می‌شود تا تعیین کنند کدام بخش از یک برچسب حافظه قطعه‌بندی شده قفل می‌شود.

Memory-Length-Encoding ۳۵-۴-۷

متغیر فرمان **Memory-Length-Encoding** در یک دستور به Data processor آگاهی می‌دهد که ظرفیت حافظه یا طول داده‌های کد شده یا هر دو را برحسب تعداد بلوک‌ها کدبندی کند. متغیر **Memory-Length-Encoding** در یک پاسخ نشان می‌دهد که Data processor طول مناسب را محاسبه کرده است و این را به عنوان بخشی از **DSFID** تعمیم یافته کدبندی کرده است.

Memory-Segment ۳۶-۴-۷

متغیر فرمان **Memory-Segment** استفاده می‌شود تا تعیین کند کدام بخش از یک برچسب حافظه قطعه‌بندی شده بر روی کدام داده کدبندی می‌شود. این شبیه به کارکرد در متغیر **Memory-Bank** می‌باشد (به زیربند ۷-۴-۳۳ مراجعه کنید) به جز برچسب RFID که برای آن این متغیر مناسب است که می‌تواند چند قطعه را در تراکنش‌های واسط هوایی مشابه، آدرس‌دهی کند.

Memory-Type ۳۷-۴-۷

متغیر فرمان **Memory-Type** استفاده می‌شود تا تعیین کنند کدام ساختار حافظه برای کدبندی یک **Monomorphic-UII** می‌باشد، که بطور مؤثرتری به Data processor دستور می‌دهد که کدام یک از متغیرهای اختیاری و شرطی و فرآیندها برای فرآیند کدبندی به کار برده شود.

Multiple-Records-Directory-Length ۳۸-۴-۷

متغیر پاسخ **Multiple-Records-Directory-Length** یک مقدار EBV-8 می‌باشد.

Multiple-Records-Features-Indicator ۳۹-۴-۷

متغیر دستور و پاسخ **Multiple-Records-Features-Indicator** یک نقشه بیت با ساختار زیر است:

- بیت ۸ با مقدار $۱_۲$ نشان می‌دهد که تمام رکوردها از **Access-Method** مشابه استفاده می‌کنند. اگر این بیت = ۰ باشد، سپس بیت‌های ۷ تا ۴ = $۰۰۰۰_۲$ می‌باشد.
- بیت‌های ۷ تا ۴، **Access-Method** ویژه را شناسایی می‌کنند اگر بطور ثابت به کار برده شود. اگر بیت ۸ = ۰ باشد، رشته مجاز برای بیت‌های ۷ تا ۴ در حال حاضر اینها هستند:

0000 نشان می‌دهد که تمام رکوردها از **No-Directory Access-Method** استفاده می‌کنند.

0001 مجاز نمی‌باشد.

0010 نشان می‌دهد که تمام رکوردها از **Packed-Objects Access-Method** استفاده کنند.

0011 نشان می‌دهد که تمام رکوردها **Tag-Data-Profile Access-Method** استفاده می‌کنند.

0100 مجاز نمی‌باشد.

0101 تا ۱۱۱۱ در حال حاضر برای **Access-Methods** اضافی ذخیره می‌شود.

- بیت ۳ با مقدار ۱ نشان می‌دهد که بعضی از رکوردها در یک رابطه سلسله مراتبی با رکوردهای دیگر هستند.
- بیت ۲ مشخص می‌کند که آیا تعداد رشته رکوردها به‌طورکامل نگهداری می‌شود (مثلا، هر بار که یک رکورد جدید اضافه می‌شود، به‌روزرسانی می‌شود) یا اینکه آیا تعداد رکوردها ممکن است مقدار جاری صحیح نباشد. مقدار ۱ نشان می‌دهد که تعداد رکوردها صحیح است، مقدار ۰ نشان می‌دهد که تعداد رشته رکوردها ممکن است غیرصحیح باشد.
- بیت ۱ با مقدار ۱ نشان می‌دهد که یک بایت اضافی وجود دارد که ویژگی‌های اضافی را نشان می‌دهد. در حال حاضر این RFU است و بنابراین به مقدار ۰ تنظیم می‌شود.

۷-۴-۴۰ NSI-Bits

متغیر فرمان **NSI-Bits** یک کد ۹- بیتی است که توسط EPCglobal تعریف می‌شود و هنگامی‌که در برچسب RFID مربوط کدبندی می‌شود، به عنوان یک پیشوند در EPC-Code می‌باشد.

۷-۴-۴۱ Number-In-Data-Element-List

متغیر فرمان **Number-In-Data-Element-List** تنها در یک رکورد سلسله مراتبی استفاده می‌شود که یک فهرست اقلام داده است. این مقدار بیانگر تعداد نمونه‌های اقلام داده‌ها می‌باشد که کدبندی می‌شود.

۷-۴-۴۲ Number-Of-Records

متغیر پاسخ **Number-Of-Records** در یکی از دو روش استفاده می‌شود. اگر بیت ۲ از **Multiple-Records-Features-Indicator**، معادل باشد با:

۰ سپس این فیلد نگهداری نمی‌شود و مقدار **Number-of-Records** باید صفر باشد، اما هر مقدار دیگر باید صرف‌نظر شود. برای مثال اگر تعدادی شمارش رکورد بطور اولیه نگهداری شود اما بعد تصمیم گرفته می‌شود که این تابع باید متوقف شود.

۱ هنگامی‌که رکوردهای جدید اضافه می‌شوند، سپس این فیلد به‌طورکامل حفظ می‌شود. زمانی که MR-header ایجاد می‌شود، مقدار صفر برای **Number-of-Records** کدبندی می‌شود.

۷-۴-۴۳ Number-Of-Tags

متغیر فرمان **Number-Of-Tags** استفاده می‌شود تا یک حدی را بر روی **Identify-Method** ویژه تعیین کنند. این یک مقدار صحیح در دامنه ۰ تا ۶۵۵۳۵ می‌باشد.

۷-۴-۴۴ Object-Offsets-Multiplier

متغیر فرمان **Object-Offsets-Multiplier** اندازه حافظه را در بیت‌ها تعریف می‌کند که برای ذخیره‌سازی آفست‌های شی، درخواست می‌شود زمانی که پارامتر **Directory-Type** در متغیر **Packed-Objects-Constructs**، آفست **Packed-Object** می‌باشد. پیاده‌سازی باید از پارامتر برای اندازه‌گیری مناسب ساختار Auxmap در Packed Object استفاده کند.

۷-۴-۴۵ Packed-Object-Directory-Type

متغیر فرمان **Packed-Object-Directory-Type** تعیین می‌کند که آیا Packed Object یک فهرست راهنما است و در این حالت کدام نوع فهرست راهنمای Packed Object ساخته شده است. یک فهرست راهنمای حضور/ غیاب، یک نقشه بیت از مقادیر ID را ارائه می‌کند که روی هر یک از Packed Objects در برچسب RFID کدبندی شده‌اند، اما هیچ نشانه‌ای از محل کدبندی داده‌ها وجود ندارد. فهرست راهنمای فیلد اندیس مقدار ترتیبی Packed Object را اضافه می‌کند که شامل مقدار ID ویژه می‌باشد. برای مثال، یک فیلد اندیس ۳- بیتی مشخص می‌کند که کدام یک از هشت Packed Objects در دسترس است. یک فهرست راهنمای آفست، آدرس شروع Packed Object را ارائه می‌کند که شامل یک مقدار ID ویژه می‌باشد.

یک فهرست راهنمای پوچ[1] (یعنی شرط فضا برای یک فهرست راهنما) با تنظیم تخصیص اشاره‌گر در این متغیر ساخته می‌شود که یک مقدار غیرصفر را برای متغیر **PO-Directory-Size** ترکیب کرده است.

اگر یک Packed Objects یک فهرست راهنما نباشد، سپس یک مقدار صفر برای این متغیر تعیین می‌شود.

۷-۴-۴۶ Password

متغیر فرمان **Password** رشته بایت را تعریف می‌کند که یک مقدار رمز را نشان می‌دهد که با **Password-Type** تعیین شده است، در یک دستوری که لازم است مطابق با رمز تعریف شده مشابه در برچسب

1 - Null

RFID باشد. اگر مطابقت داشته باشد، مجوزهایی را فراهم می‌کند تا عمل‌های اضافی را با برچسب RFID انجام دهند. یک عدم‌انطباق موجب رد شدن هر دستور مربوطه می‌شود.

Password-Type ۴۷-۴-۷

متغیر فرمان **Password-Type** مقدار **Password** را تعیین می‌کند تا مجوزهایی را تهیه کنند تا عمل‌های اضافی را با برچسب RFID انجام دهند.

PO-Directory-Size ۴۸-۴-۷

متغیر فرمان **PO-Directory-Size** اندازه اشاره‌گر فهرست راهنمای (پوچ) را تعریف می‌کند که با Packed Objects ساخته شده است. این متغیر فرمان به صورت یک مقدار صحیح غیرصفر ارائه می‌شود که باید برای اندازه اشاره‌گر فهرست راهنما (پوچ) استفاده شود که در Packed Objects رمز شده است.

PO-Index-Length ۴۹-۴-۷

متغیر فرمان **PO-Index-Length** استفاده می‌شود تا اندازه POindex Length را برای ساختار AuxMap تعیین کنند تا هنگامی ساخته شود که پارامتر **Directory-Type** Packed Objects، فیلد اندیس Packed Objects باشد.

Pointer ۵۰-۴-۷

متغیر فرمان **Pointer** در ترکیب عطفی[1] با متغیرهای فرمان **Length-of-Mask** و **Tag-Mask** استفاده می‌شود تا معیار جستجو را از داده‌های EPCglobal تعریف کنند که در حافظه UII یک برچسب RFID حافظه قطعه‌بندی شده، کدبندی شده‌اند.

Pointer-To-Multiple-Record-Directory ۵۱-۴-۷

متغیر فرمان و پاسخ **Pointer-To-Multiple-Record-Directory** استفاده می‌شود تا بیشترین تعداد بلوک را در آدرس شروع فهرست راهنما تعریف کنند. این به عنوان یک مقدار EBV-8 تعریف می‌شود. اگر یک فهرست راهنما در ابتدا رمز نشود، آنگاه یک رشته EBV-8 با طول ثابت (که این طول معادل مقدار مورد نیاز برای نقطه شروع فهرست راهنما است)، باید با یک مقدار صفر کدبندی شود. این فهرست راهنما در ترتیب بلوک معکوس رمز می‌شود؛ بطوریکه بلوک بعدی فهرست راهنما در آدرس پایین‌تر بعدی است. لزوما شروع فهرست راهنما بالاترین آدرس در Logical Memory نمی‌باشد، زیرا ویژگی‌های سخت‌افزاری دیگر برچسب ممکن است در آنجا تعیین موقعیت شوند. لازم است که دستورات واسط هوایی فراخوانی شود. این دستورات نگاشت حافظه را برای تعیین آدرس شروع فهرست راهنما، تعریف می‌کند.

1 - Conjunction

۷-۴-۵۲ Read-Record-Type

متغیر فرمان Read-Record-Type برای شناسایی ساختارهای منطقی مختلف از برچسب RFID استفاده می‌شود. این متغیر فرمان به صورت یک عدد صحیح ارائه می‌شود و کدهای زیر به کار برده می‌شوند:

۰	**Read-Multiple-Records-Header**
۱	**Read-Multiple-Records-Header-Plus-1st-Preamble**
۲	**Read-Multiple-Records-Directory**
۳	**Read-Preamble-Specific-Multiple-Record**
۴	**Read-All-Record-OIDs-Specific-Record-Type**
۵	**Read-OIDs-Specific-Multiple-Record**
۶	**Read-All-Objects-Specific-Multiple-Record**
۷	**Read-Multiple-Objects-Specific-Multiple-Record**
۸	**Read-1st-Objects-Specific-Multiple-Record**
۹	**Read-Data-Element-List-Specific-Multiple-Record**

اگر **Read-Multiple-Records-Header** انتخاب شود، Data Processor، تفسیر MR-header را در متغیر **Multiple-Records-Header-Structure** بر می‌گرداند.

اگر **Read-Multiple-Records-Header-Plus1st-Preamble** انتخاب شود، Data Processor، تفسیر MR-header را در متغیر **Multiple-Records-Header-Structure** و تفسیر اولین مقدمه رکورد در متغیر **Multiple-Records-Preamble-Structure** بر می‌گرداند.

اگر **Records-Multiple-Records-Directory** انتخاب شود، Data processor، تفسیر چند فهرست راهنمای رکوردها را در متغیر **Multiple-Records-Directory-Structure** بر می‌گرداند.

کدهای ۳ تا ۹ Read-Record-Type باید از یک یا چند Object-Identifier در دستور برای Data processor استفاده کنند تا فرآیندهای مناسب را درخواست کنند. سه قالب بسیار ویژه از **Object-Identifier** در چند رکورد به کار برده می‌شوند:

- برای یک رکورد چندگانه که بخشی از یک سلسله مراتب نمی‌باشد، ساختار بصورت زیر است:

1.0.15961.401.{Data-Format}.{sector identifier}.{record type}.{instance-of}.{Relative-OID of data element}

- برای یک رکورد چندگانه که بخشی از یک سلسله مراتب است، اما نه یک فهرست اقلام داده، ساختار بصورت زیر است:

1.0.15961.402.{Data-Format}.{sector identifier}.{record type}.{hierarchical id}.{Relative-OID of element}

- برای یک رکورد چندگانه که فهرست اقلام داده است، ساختار بصورت زیر است:

1.0.15961.403.{Data-Format}.{sector identifier}.{record type}.{hierarchical id}.{data element}

یادآوری- این شناسه شی پاسخی را درخواست می‌کند که شامل تمام تعداد اقلام فهرست می‌باشد.

دو ساختار اول **Object-Identifier** در کدهای ۳ تا ۸ **Read-Record-Type** به کار برده می‌شوند. سومین ساختار **Object-Identifier** فهرست شده تنها در کدهای ۳، ۴ و ۹ **Read-Record-Type** به کار برده می‌شود. اگر **Read-Preamble-Specific-Multiple-Record** انتخاب شود، این دستور باید شامل یک **Object-Identifier** واحد در **Read-Objects-List** باشد که به‌طور اطمینان‌بخش پایین قوس نوع رکورد و نمونه قوس (اگر کاربردی باشد) یا قوس شناسه سلسله مراتبی (اگر کاربردی باشد) تعریف شده است. Data Processor تفسیر مقدمه رکورد را در متغیر **Multiple-Records-Preamble-Structure** بر می‌گرداند.

انتخاب **Read-All-Record-OIDs-Specific-Record-Type** برای شناسایی مجموعه رکوردهای تاریخی از نوع مشابه یا یک مجموعه رکوردها در یک سلسله مراتب از نوع مشابه، مفید می‌باشد. در این حالت دستور باید شامل یک **Object-Identifier** در **Read-Objects-List** باشد که تنها در **Read-OIDs-Response-List** تعریف می‌شود. Data Processor فهرست یک لایه **Object-Identifier** پایین‌تر را در **Read-OIDs-Response-List** بر می‌گرداند، یعنی با مجموعه‌ای از نمونه قوس‌ها یا با مجموعه‌ای از قوس‌های شناسه سلسله مراتبی.

اگر **Read-OIDs-Specific-Multiple Response** انتخاب شود، دستور باید شامل یک **Object-Identifier** در **Read-Object-list** باشد که به‌طور اطمینان‌بخش در قوس نوع رکورد و نمونه قوس (اگر کاربردی باشد) یا قوس شناسه سلسله مراتبی (اگر کاربردی باشد) تعریف می‌شود. **Data Processor** فهرست **Object-Identifier** کد شده در رکورد را در **Read-OIDs- Response-List** بر می‌گرداند. این در فهرست‌های Data Element استفاده نمی‌شود.

اگر **Read-All-Objects-Specific-Multiple-Record** انتخاب شود، این دستور باید شامل یک **Object-Identifier** در **Read-Object-list** باشد که به‌طور اطمینان‌بخش در قوس نوع رکورد و نمونه قوس (اگر کاربردی باشد) یا قوس شناسه مرتبه‌ای (اگر کاربردی باشد) تعریف شده است. Data Processor فهرست **Object-Identifier** و **Objects** رمز شده در رکورد را در **Read-Objects-Response-List** بر می‌گرداند. این در فهرست‌های Data Element استفاده نمی‌شود.

اگر **Read-Multiple-Objects-Specific-Multiple-Record** انتخاب شود، دستور باید شامل **Object-Identifier** معرفی شده باشد که در اقلام داده‌های ویژه در **Read- Object- list** تعریف می‌شوند. Data Processor فهرست **Object-Identifier** معرفی شده و Objects رمز شده درون رکورد را در **Read-Objects-Response-List** بر می‌گرداند. این در فهرست‌های Data Element استفاده نمی‌شود.

اگر **Read-1st-Objects-Specific-Multiple-Record** انتخاب شود، دستور باید شامل **Object-Identifier** معرفی شده باشد که در اقلام داده‌های ویژه در **Read-Object-List** تعریف می‌شوند. Data Processor فهرست **Object-Identifier** معرفی شده و Objects رمز شده به **Max-App-Length** درون رکورد را در **Read-Objects-Response-List** بر می‌گرداند. این در فهرست‌های Data Element استفاده نمی‌شود.
اگر **Read-Data-Element-List-Specific-Multiple-Record** انتخاب شود، دستور باید شامل یک **Object-Identifier** در **Read-Object-list** باشد که در اقلام داده تعریف می‌شود. Data Processor ابتدا بررسی می‌کند که رکورد یک فهرست Data Element است. اگر این‌طور باشد، از قواعد **Access-Method** استفاده می‌کند تا **Object-Identifier** را درتعداد اقلام فهرست بازسازی کند همان‌طور که در فهرست اقلام داده رمز شده است و این را بر می‌گرداند و **Objects** مربوطه درون رکورد در **Read-Objects-Response-List** رمز شده‌اند.

۷-۴-۵۳ Read-Type

Read-Type یک متغیر فرمان است که تعداد و موقعیت **Object-Identifier** را در یک دستور خواندن شناسایی می‌کند. این متغیر فرمان به صورت یک مقدار صحیح ارائه می‌شود و کدهای زیر استفاده می‌شوند:

۰	**Read-1st-Objects**
۱	**Read-Multiple-Objects**
۲	**Read-All-Objects**
۳	**Read-Monomorphic-UII**
۴ تا ۱۵	برای تعریف آینده کنار گذاشته شده است.

اگر **Read-1st-Objects** انتخاب شود، سپس دستور باید شامل مقداری برای کد **Max-App-Length** باشد که معادل با تعداد بایت‌هایی می‌باشد که در برنامه کاربردی خوانده می‌شود. این متغیر بسیار ساختاری است و بنابراین از استاندارد ISO/IEC 15961: 2004 - تا یک ترتیب **Object-Identifiers** را بخوانند، به جای اینکه تنها در اولین موقعیت باشند- متفاوت می‌باشد. اگر **Read-Type** مجموعه‌ای از **Read-Multiple-Objects** باشد، این می‌تواند در یک یا چند **Object-Identifiers** استفاده شود.
اگر **Read-Monomorphic-UII** انتخاب شود، به Data processor دستور می‌دهد که بررسی‌های اضافی را با ثبت دستورات ISO/IEC Data انجام دهند تا فرآیند مناسب را در دستور فعال کنند.

۷-۴-۵۴ Read-Memory-Capacity

متغیر فرمان **Read-Memory-Capacity** در فرمان‌ها استفاده می‌شود تا یک رکورد چندگانه را بنویسید تا مقدار حافظه‌ای (برحسب بلوک‌های نوشتن) که باید تخصیص داده شود را نشان دهند. به‌طورمعمول رکورد را

فعال می‌کنند تا اقلام داده‌های اضافی را داشته باشند. تنها زمانی استفاده می‌شود که نیاز است برنامه کاربردی، اندازه خودکار را توسط Data Processor لغو کند، زمانی که متغیر **Application-Defined-Record-Capacity** به TRUE تنظیم می‌شود.

۷-۴-۵۵ Record-Type-Arc

متغیر پاسخ **Record-Type-Arc** مقداری صحیح است که از مقدار EBV-8 رمز شده در مقدمه Multiple Records، برگردانده شده است.

۷-۴-۵۶ Record-Type-Classification

متغیر فرمان و پاسخ **Record-Type-Classification** رشته بیتی است که کلاس رکوردی را شناسایی می‌کند که کد می‌شود. کدهای زیر استفاده می‌شوند:

۰۰۰ رکورد مستقل با instance-of arct = 0

۰۰۱ رکورد مستقل با instance-of arct > 0

۰۱۰ رکورد سلسله مراتبی، سطح بالا

۰۱۱ رکورد سلسله مراتبی، که هم والد و فرزند(ان) دارد

۱۰۰ رکورد سلسله مراتبی، فهرست راهنمای اقلام داده

۱۰۱ دیگر رکورد سلسله مراتبی، بدون فرزند (برای فهرست راهنمای اقلام داده به کار نمی‌رود)

۱۱۰ با این فرمان مرتبط نمی‌باشد (زیرا با رکوردهای حذف شده مرتبط است)

۱۱۱ ذخیره شده است

۷-۴-۵۷ Sector-Identifier

Sector-Identifier در MR-header استفاده می‌شود تا **Sector-Identifier** صحیح را برای تمام رکوردها در Logical Memory نشان دهند و یا در سیگنالی نشان دهند که مقدار صحیح بین رکوردها متغیر است و مقدار صحیح در رکورد رمز می‌شود. این متغیر فرمان و پاسخ به عنوان یک مقدار صحیح ارائه می‌شود و کدهای زیر استفاده می‌شوند:

۰ **Sector-Identifier** صحیح، در هر رکورد کدبندی می‌شود و می‌تواند بین رکوردها متفاوت باشد.

۱ این برای کاربردهای سیستم بسته استفاده شود.

۲ این نشان می‌دهد که نوع رکورد، مقداری برابر با **Relative-OID** اولین عنصر داده دارد که روی رکورد کدبندی شده است. برای مثال، اگر اولین عنصر داده، برای یک کد محصول، یک **Relative-OID** = 7 داشته باشد، آنگاه نوع رکورد = ۷ است.

>۲ **Sector-Identifier** توسط مدیران واژه‌نامه داده به یک بخش داده شده است تا انواع رکورد تخصیص خودشان را مدیریت کند.

یادآوری- اگر تمام رکوردها، **Sector-Identifier** همانند داشته باشند، مقدار غیر صفر بخشی از ساختار **Object-Identifier** در یک دستور می‌باشد تا یک رکورد را بنویسند یا بخوانند، اما باید برای دستور **Configure-Multiple-Records-Header** جداگانه قرار گیرند.

۷-۴-۵۸ Simple-Sensor-Indicator

متغیر فرمان **Simple-Sensor-Indicator** توسط برنامه کاربردی استفاده می‌شود که برچسب RFID، از یک حسگر ساده پشتیبانی می‌کند. این متغیر بطور کلی زمانی استفاده می‌شود که پروتکل واسط هوایی، یک ویژگی برای پشتیبانی از چنین نشانه‌ای ندارد.

۷-۴-۵۹ Start-Address-Of-Record

این متغیر پاسخ، آدرس اولین بایت Multiple Record کدبندی شده را برحسب اندازه بلوک نوشتن فراهم می‌کند، همان‌طور که در قالب EBV-8 درچند فهرستی رکوردها کدبندی شده است.

۷-۴-۶۰ Tag-Data-Profile-ID-Table

Tag-Data-Profile-ID-Table یک متغیر فرمان است که برای شناسایی **Tag-Data-Protocol** ویژه، استفاده شده است که شامل قواعد فشرده‌سازی و قالب‌بندی می‌باشد. قواعد خاصی برای کدبندی هر داده در یک برنامه کاربردی معین با استفاده از این **Access-Method** لازم هستند.

۷-۴-۶۱ Tag-Mask

متغیر فرمان **Tag-Mask** درترکیب عطفی با متغیرهای فرمان **Length-of-Mask** و **Pointer** استفاده می‌شود تا معیار جستجوی داده‌های EPCglobal را در حافظه UII از یک برچسب RFID حافظه قطعه‌بندی شده، تعریف کند.

۷-۴-۶۲ Update-Multiple-Records-Directory

متغیر فرمان **Update-Multiple-Records-Directory**، BOOLEAN است، اما پردازش بر روی Logical Memory توسط Data Processor نیاز دارد که در نظر گرفته شود که آیا یک فهرست وجود دارد. وضعیت‌ها و فرآیندهای زیر به کاربرده می‌شوند:

- اگر یک فهرست از قبل وجود داشته باشد و متغیر فرمان به TRUE تنظیم شود، این فهرست به‌طورکامل به‌روزرسانی می‌شود، از جمله هر ورودی فهرست که از قبل از بین رفته است.
- اگر یک فهرست از قبل وجود داشته باشد و متغیر فرمان به FALSE تنظیم شود، این متغیر نادیده گرفته می‌شود و فهرست به‌طورکامل به‌روزرسانی می‌شود، از جمله هر ورودی فهرست که از قبل از بین رفته است.
- اگر هیچ فهرستیی وجود نداشته باشد و متغیر فرمان به TRUE تنظیم شود، این فهرست ایجاد می‌شود و به‌طورکامل به روز است، از جمله هر ورودی فهرست که از قبل از بین رفته است.
- اگر هیچ فهرستیی وجود نداشته باشد و متغیر فرمان به FALSE تنظیم شود، هیچ فهرستی ساخته نمی‌شود.

Word-Count ۶۳-۴-۷

متغیر **Word-Count** در ترکیب عطفی با **Word-Pointer** استفاده می‌شود تا تعداد بایت‌های کد شده را تعریف کند که از یک برچسب RFID حافظه قطعه‌بندی شده خوانده می‌شود، بدون اینکه Data Processor پرداشی انجام دهد.

Word-Pointer ۶۴-۴-۷

متغیر فرمان **Word-Pointer** در ترکیب عطفی با **Word-Count** استفاده می‌شود و موقعیت شروع حافظه یک برچسب RFID حافظه قطعه‌بندی شده را تعریف می‌کند که از آن یک رشته بایت کدبندی شده خوانده می‌شود.

۵-۷ نام‌های فیلد مربوط به فرمان

علاوه بر متغیرهای فرمان (زیربند ۷-۴)، نام فیلدهای زیر در فرمان‌ها و پاسخ‌ها استفاده می‌شوند.

Data-Set ۱-۵-۷

نام فیلد Data-Set به رشته بابت پیوسته از **Object-Identifier, Precursor** در بایت‌های نهایی **Object** فشرده روی یک برچسب RFID اشاره می‌کند.

Identities ۲-۵-۷

نام فیلد **Identities**، یک فیلد پاسخ فرمان است که فهرستی **Singulation-Ids** مشخص‌شده، را نمایش می‌دهد.

Length-Lock Byte ۳-۵-۷

نام فیلد **Length-Lock Byte** یک فیلد پاسخ فرمان است که طول داده‌های کدشده را در یک قطعه مرتبط با اقلام ، شناسایی می‌کند و همچنین اطلاعاتی را در مورد وضعیت قفل صفحه در برچسب RFID تهیه می‌کند.

Length-of-Encoded-Data ۴-۵-۷

نام فیلد **Length-of-EnCoded-Data** یک فیلد پاسخ فرمان است که بخشی از **DSFID** تعمیم یافته می‌باشد و طول داده‌های کد شده را برحسب بلوک‌ها مشخص می‌کند.

Lock-Status ۵-۵-۷

نام فیلد **Lock-Status** یک فیلد پاسخ فرمان است که شناسایی می‌کند آیا یک بسته کدبندی (مثلا یک **Data-set**) قفل شده است یا نه.

Logical-Memory-Map ۶-۵-۷

فیلد **Logical-Memory-Map** یک فیلد پاسخ فرمان است که رشته بایت کامل اما کدبرداری نشده را نشان می‌دهد که از برچسب RFID خوانده شده است.

۷-۵-۷ Memory-Capacity

نام فیلد **Memory-Capacity** یک رشته پاسخ فرمان است که بخشی از **DSFID** تعمیم یافته می‌باشد و طول داده‌های کدبندی شده را برحسب بلوک‌ها شناسایی می‌کند.

۷-۵-۸ Module-OID

فیلد Module-OID یک پودمان فرمان یا پاسخ فردی را با یک **Object-Identifier** کامل شناسایی می‌کند همان‌طور که در زیربند ۷-۳-۲ تعریف شده است.

۷-۵-۹ Number-Of-Tags-Found

نام فیلد **Number-Of-Tags-Found** یک فیلد پاسخ فرمان است که تعداد واقعی برچسب‌های RFID مشاهده شده را بر می‌گرداند که مطابق با معیار بوده است. اگر متغیر **Identify-Method** در **Inventory-No-More-Than** قرار گیرد، آنگاه مقدار پاسخ **Number-of-Tags-Found** ممکن است یک عدد پایین‌تری باشد.

۷-۵-۱۰ PO-ID-Table

نام فیلد **PO-ID-Table** یک فیلد فرمان است که جدول خاصی را برای Data Processor شناسایی می‌کند تا به‌عنوان منبع قواعد کدبندی همراه با جزئیات استفاده شود.

۷-۵-۱۱ Protocol-Control-Word

نام فیلد **Protocol-Control-Word** یک فیلد پاسخ فرمان است که این مقدار ۱۶ بیتی را در یک پاسخ از حافظه UII از یک برچسب RFID حافظه قطعه‌بندی شده، بر می‌گرداند. **Protocol-Control-Word** شامل داده‌های بیتی است که دیگر مشخصه‌های پشتیبانی شده بر روی برچسب RFID را شناسایی می‌کند.

۷-۵-۱۲ Read-Data

فیلد Read-Data یک رشته بایت رمزگشایی نشده[1] را از یک حافظه قطعه‌بندی شده برچسب‌های RFID بر می‌گرداند.

۷-۶ امنیت داده‌ها

داده‌ها (**Object**) ممکن است با استفاده از چند شکل رمزنگاری ایمن شوند[2]. این باید قبل از انتقال **Object** به Data Processor انجام شود. فرآیند کشف رمز نیز باید در **Object** به کار برده شود پس ازآنکه به برنامه کاربردی منتقل شده است. همین‌طور، تمام فرآیندها دراین استاندارد و استاندارد ISO/IEC 15962 شفاف هستند.

ویژگی‌های زیر می‌توان تهیه کرد:

1 - Un-decoded
2 - Made Secure

- امنیت داده‌ها - تنها کاربران مجاز می‌توانند داده‌های صحیح را «ببینند». فرستنده باید الگوریتم کشف رمز و کلید را برای کاربران مجاز ارائه دهد. کاربران دیگر می‌توانند رشته هشتایی پردازش‌نشده را ببینند اما این بی‌معنی خواهد بود.
- یکپارچگی داده‌ها - از داده‌های کشف کد شده استفاده می‌کند تا مشخص کند که آیا داده‌ها قبل از خوانده شدن توسط یک کاربر مجاز اصلاح شده‌اند.
- اعتبار داده‌ها - از فرآیندهای رمزدار کردن استفاده می‌کند تا ریسک نسخه‌برداری شدن داده‌ها را از یک منبع قانونی توسط یک نماینده غیرمجاز در برچسب RFID دیگر کاهش دهد و به عنوان برچسب RFID منبع اصلی نادیده گرفته شود.

توصیه‌های دیگر در پیوست ت ارائه می‌شود.

۸ جریان‌های داده و فرآیندها در واسط هوایی

فرآیندهای مختلف لازم هستند تا برچسب RFID را قالب‌بندی کنند، داده‌ها را در آن بنویسند، از آن بخوانند، داده‌ها را اصلاح کنند و غیره. اینها در زیربندهایی تعریف می‌شوند که در قسمت زیر مطرح می‌شوند. تمام فرآیندها برای نوشتن و افزودن داده‌ها توصیف خواهند شد. اگر فرآیند خواندن عکس فرآیند نوشتن باشد، بطور خلاصه توصیف خواهد شد، در غیراین‌صورت یک شرح دیگری تهیه خواهد شد.

۸-۱ برقراری ارتباطات بین برنامه کاربردی و برچسب RFID

Data Processor ارتباط مستقیمی با برچسب RFID ندارد (به شکل ۱- نمودار لایه‌های پروتکل برای یک پیاده‌سازی RFID برای مدیریت اقلام و شکل ۲- توابع منطقی و واسط‌های استاندارد ISO/IEC 15962 با مولفه‌های سامانه RFID دیگر- مراجعه کنید)، اما این را از طریق Tag Driver انجام می‌دهد. Data Processor به اطلاعات سامانه ویژه براساس پیکربندی برچسب RFID نیاز دارد (به زیربند ۸-۱-۲ مراجعه کنید). این، پارامتر Logical Memory را تنظیم می‌کند تا به‌طور مستقیم حافظه برچسب RFID را نشان دهد و ارتباطات را در Tag Driver فعال کند. برای دستیابی به این، خدمات واسط هوایی باید در Data Processor از طریق Tag Driver تهیه شوند تا ارتباطات را برقرار کنند (به زیربند ۸-۱-۱ مراجعه کنید).

تعدادی از این پارامترها باید به Data Processor شناسانده شوند یا توسط Data Processor درخواست شوند. بطور مؤثر، این رویه در ابتدا برچسب RFID را پیکربندی می‌کند تا در شرایط لازم آن را دوباره پیکربندی کند و در حالی‌که تراکنش داده‌ها باز است، یک پیوند ارتباطات را برقرار کند.

۸-۱-۱ خدمات واسط هوایی

این استاندارد با توجه به این واقعیت که انواع جدیدی از برچسب RFID ممکن است به استاندارد ISO/IEC 18000 اضافه شود و Data Processor را بدون تغییر بگذارد، انتها- باز[1] است. برای دستیابی به این، چند پیش‌فرض اصلی در مورد انواع برچسب RFID در استاندارد ISO/IEC 18000 ساخته می‌شود.

- حافظه کاربردی، عدد صحیحی از بایت‌ها می‌باشد.

یادآوری- عبارت حافظه کاربردی در این زیربند به عنوان یک نام کلی برای ناحیه حافظه برچسب RFID استفاده می‌شود که برای داده کاربر (در استاندارد ISO/IEC 18000، گاهی اوقات حافظه کاربر نامیده می‌شود)در دسترس می‌باشد و هر حافظه جداگانه قابل تشخیص است (مثلا برای UII).

- حافظه کاربردی باید در بلوک‌ها تشکیل شود. اینها باید در اندازه ثابت باشند و یک یا چند بایت داشته باشند.
- هر یک از بلوک‌ها باید توسط خواندن و یا نوشتن قابل دسترس باشند.

یادآوری- این، برای تابع اولیه به کار برده می‌شود، مشخصات اضافی ممکن است استفاده شوند تا دسترسی به کاربران مجاز را محدود کنند.

- علاوه بر الزامات (بالا) مربوط به حافظه، یک سازوکار قابل اطمینان برای نوشتن بر و خواندن از حافظه کاربردی باید وجود داشته باشد.

برچسب RFID باید یک سازوکار برای ذخیره‌سازی اطلاعات سامانه داشته باشد (به زیربند ۸-۱-۲ مراجعه کنید) از جمله توانایی نوشتن و خواندن اقلام مولفه.

جزئیات فنی خدمات واسط هوایی در استاندارد ISO/IEC 15962 تهیه می‌شوند.

۸-۱-۲ اطلاعات سامانه

اطلاعات سامانه باید شامل اقلام زیر باشد که باید در واسط کاربردی و واسط هوایی منتقل شوند و بنابراین بخشی این استاندارد و استاندارد ISO/IEC 15962 می‌باشند: **Singulation-Id** - (به زیربند ۷ -۲-۱ مراجعه کنید).

AFI - (به زیربند ۷-۲-۲ مراجعه کنید)

DSFID(به زیربند ۷-۲-۳ مراجعه کنید) که خود شامل:

Access-Method - (به زیربند ۷-۲-۴ مراجعه کنید

Data-Format- (به زیربند ۷-۲-۵ مراجعه کنید)

اطلاعات سامانه نیز باید شامل اقلام زیر باشد که تنها باید بین برچسب RFID و Data Processor منتقل شود تا Logical Memory Map را پیکربندی کند و بنابراین تنها بخشی از استاندارد ISO/IEC 15962 می‌باشند:

---

1 - Open-ended

- اندازه بلوک فیزیکی
- تعداد بلوک‌ها

۸-۲ خدمات سامانه برنامه کاربردی

سامانه برنامه کاربردی باید موارد زیر را تهیه کند:

- **Object Identifier** (به زیربند ۷-۳-۲ مراجعه کنید)
- **Object** (به زیربند ۷-۳-۴ مراجعه کنید)
- **Compact-Parameter** (به زیربند ۷-۳-۵- مراجعه کنید)
- **Object-Lock** (به زیربند ۷-۳-۶ مراجعه کنید)

اینها در تعریف‌های فرمان‌ها و پاسخ‌های کاربردی با هم ترکیب می‌شوند و بدون فرمان‌های پشتیبانی، هرگز منتقل نمی‌شوند.

۹ Command-Codes، Completion-Codes و Execution-Codes

فرمان‌های برنامه کاربردی برای آموزش Data Processor و تحقیق‌کننده استفاده می‌شوند تا کارکردهای خاصی را اجرا کنند. آن‌ها همچنین در پردازش داده‌های کاربردی به کار برده می‌شوند تا کدبندی خوبی را انجام دهند. پاسخ‌ها از Data Processor که شامل داده‌های تقاضا شده و همچنین اطلاعاتی در مورد عمل[1]های تقبل شده و خطاها است، پیدا شده است. هر جفت فرمان/ پاسخ، قواعد نحوی انتزاعی خود را دارد که به صورت پودمان‌ها نشان داده شده‌اند. هر پودمان طوری تعریف می‌شود که هر فرمانی می‌تواند جدا از هر فرمان دیگری فراخوانی شود.

بنابراین این فرمان‌ها و پاسخ‌ها را می‌توان به راحتی در قواعد نحوی انتقال، ترکیب کرد، مقادیر کد در این استاندارد به قوس نهایی فرمان و پودمان‌های پاسخ تخصیص داده شده‌اند (به زیربند ۹-۱ مراجعه کنید). همچنین، **Completion-Codes** (به زیربند ۹-۲ مراجعه کنید) و **Execution-Codes** (به زیربند ۹-۳ مراجعه کنید) به پاسخ‌ها تخصیص داده می‌شوند. منابع تعریف‌های متغیرها و فیلدهایی که در هر فرمان اجرا می‌شوند ارائه می‌شود، از جمله آن‌هایی که تنها در پودمان‌های فرمان و پاسخ به کاربرده می‌شوند (به زیربندهای ۷-۴ و ۷-۵ مراجعه کنید).

در پردازش یک فرمان، یک خطا ممکن است کشف شود. این فرمان باید صرف‌نظر شود و هیچ داده‌ای به (یا از) برچسب RFID منتقل نمی‌شود که در آن خطا کشف شده است. این امکان‌پذیر است زیرا پردازش در Logical Memory انجام می‌گیرد. اگرچه سایر شرایط خطا ممکن است وجود داشته باشد، اولین مساله‌ای که تشخیص داده شد تنها یک بار گزارش می‌شود. **Completion-Code** یا **Execution-Code** مناسب برگردانده می‌شود.

1 - Action

اگر فرمان، چندین برچسب RFID را نشان دهد، تمام برچسب‌های RFID پردازش شده قبل از کشف خطا باید پردازش شوند. کشف خطا تمام پردازش بعدی را صرف‌نظر می‌کند.

زیربندهای زیر تمام فرمان‌های کاربردی و پاسخ‌ها را تعریف می‌کنند که با این ویرایش این استاندارد پشتیبانی می‌شوند. علاوه بر قواعد نحوی اصلی، این زیربندها نیز کارکرد و هدف متغیرهای فرمان ویژه و پاسخ‌ها را توصیف می‌کنند. فرمان‌ها و پاسخ‌ها بطور منطقی با هم دسته‌بندی می‌شوند. در این بند، قواعد نحوی انتزاعی مناسب فعلی نشان داده می‌شود.

پیوست ث،۱۶ پودمان اصلی را با استفاده از قواعد انتزاعی ASN.1 نشان می‌دهد و از اصطلاحات در استاندارد ISO/IEC 15961: 2004 استفاده شده است. بعضی از این پودمان‌ها معادل‌های مستقیمی درقالب جاری دارند، بقیه ادغام شده‌اند تا پودمان‌های جدید ایجاد کنند و این در زیربندهای مناسب توصیف خواهد شد. پیوست ث یک مثال از کدبندی انتقال اصلی یک فرمان و پاسخ را نشان می‌دهد.

۹-۱ مقادیر قوس نهایی از پودمان‌های فرمان و پاسخ

هر پودمان فرمان و پاسخ باید با یک **Object-Identifier** شناسایی شود. ریشه مشترک برای فرمان‌ها **{iso (1) standard(0) rfid-data-protocol(15961) commandModules (126)}** می‌باشد. ریشه مشترک برای پاسخ این است:**{iso (1) standard (0) rfid- data- protocol (15961) commandResponses (127)}**.

قوس نهایی هر جفت از پودمان‌های فرمان و پاسخ باید هم مقدار باشند، بطور مؤثری یک Relative-OID. قوس نهایی باید برای جفت فرمان و پاسخ مشخص باشد و قوس‌های نهایی زیر تعیین شده است:

۱ **Configure-AFI** (در استاندارد ISO/IEC 15961:2004 با عنوان **configureAfi**)

۲ **Configure-DSFID** (در استاندارد ISO/IEC 15961:2004 با عنوان **configureStorageFormat**)

۳ **Inventory-Tags**

۴ (فقط در استاندارد ISO/IEC 15961:2004 با عنوان **addSingleObject**)

۵ **Delete-Object**

۶ **Modify-Object**

۷ (فقط در استاندارد ISO/IEC 15961:2004 با عنوان **readSingleObject**)

۸ **ReadObject-Identifiers** (در استاندارد ISO/IEC 15961:2004 با عنوان **readObjectlds**)

۹ (فقط در استاندارد ISO/IEC 15961:2004 با عنوان **readAllObjects**)

۱۰ **Read-Logical-Memory-Map**

۱۱ (فقط در استاندارد ISO/IEC 15961:2004 با عنوان **inventoryAndReadObjects**)

۱۲ **Erase-Memory**

۱۳ **Get-App-Based-System-Info**

۱۴ (فقط در استاندارد ISO/IEC 15961:2004 با عنوان **addMultipleObjects**)

۱۵ (فقط در استاندارد ISO/IEC 15961:2004 با عنوان **readMultipleObjects**)

۱۶ (فقط در استاندارد ISO/IEC 15961:2004 با عنوان **readFirstObjects**)

۱۷ **Write-Objects**

۱۸ **Read-Objects**

۱۹ **Write-Objects-Segmented-Memory-Tag**

۲۰ **Write EPC-UII**

۲۱ **Inventory-ISO-UIImemory**

۲۲ **Inventory-EPC-UIImemory**

۲۳ **Write-Password-Segmented-Memory-Tag**

۲۴ **Read-Words-Segmented-Memory-Tag**

۲۵ **Kill-Segmented-Memory-Tag**

۲۶ **Delete-Packed-Object**

۲۷ **Modify-Packed-Object**

۲۸ **Write-Segments-6TypeD-Tag**

۲۹ **Read-Segments-6TypeD-Tag**

۳۰ **Write-Monomorphic-UII**

۳۱ **Configure-Extended-DSFID**

۳۲ **Configure-Multiple-Records-Header**

۳۳ **Read-Multiple-Records**

۳۴ **Delete-Multiple-Record**

پودمان‌های اضافی فرمان و پاسخ و مقادیر قوس نهایی آن‌ها، در ترتیب عددی، در شرایط لازم به این استاندارد اضافه خواهند شد.

پودمان کامل تابعی را تعیین می‌کند که تحقیق‌کننده باید انجام دهد. هر فرمان در شرایط مناسب فرآیندهای ویژه را تعیین می‌کند که توسط Data Processor، Tag Driver و تحقیق‌کننده در ارتباطات در واسط هوایی انجام می‌شود. هر پاسخ در شرایط مناسب فرآیندهای خاصی را تعیین می‌کند تا توسط Tag Driver، Data Processor و ارتباطات در واسط کاربردی تقبل شود.

۹-۲ Completion-Code

Completion-Code، قسمتی از پاسخ هر فرمان است. **Completion-Code** یک مقدار INTEGER است که بطور اختصاصی گزارش می‌دهد که چگونه فرمان خاصی پردازش و اجرا شده است، با موفقیت و یا بدون موفقیت.

در هر پاسخ برگردانده می‌شود. اگر مقدار آن (00_{16}) 0 باشد، این فرمان با موفقیت اجرا شده است. اگر مقدار آن 255(FF_{16}) باشد، این فرمان را نمی‌توان بوسیله سامانه اجرا کرد به دلیلی که در **Execution-Code** مشخص شده است (به زیربند ۹-۳ مراجعه کنید). اگر مقدار آن متفاوت از ۰ و ۲۵۵ باشد، این نشان می‌دهد که فرمان طبق آموزش برنامه کاربردی به دلیلی که عنوان شد اجرا نشده است.

یادآوری- **Completion-Code** اطلاعاتی را در مورد مبنایی فراهم می‌کند که این فرمان را می‌توان برای برچسب RFID ویژه در زنجیره ارتباطات صدا زد، در حالی که **Execution-Code** یک خطا یا موفقیت سامانه‌ها را نشان می‌دهد.

Completion-Codes در قسمت زیر به کاربرده می‌شوند:

۰ **No-Error**: این فرمان با موفقیت اجرا شد.

۱ **AFI-Not-Configures**: این فرمان نمی‌تواند کامل شود، به‌طور احتمال به دلیل یک عمل پیکربندی قبلی می‌باشد.

۲ **AFI-Not-Configured-Locked**: **AFI** قفل شده است (توسط اجرای فرمان قبلی) بنابراین این AFI را نمی‌توان پیکربندی کرد، همان‌طور که با فرمان در این موقعیت درخواست شده است.

۳ **AFI-Not-Configured-Locked-Failed**: **AFI** به درستی پیکربندی و کدبندی شد، اما تابع قفل نمی‌تواند کامل شود.

۴ **DSFID-Not-Configured**: فرمان نمی‌تواند کامل شود و به‌طور احتمال به دلیل یک عمل پیکربندی قبلی است.

۵ **DSFID-Not-Configured-Locked**: **DSFID** قفل شده است (توسط اجرای فرمان قبلی) بنابراین این **DSFID** را نمی‌توان پیکربندی کرد، همان‌طور که با فرمان در این موقعیت درخواست شده است.

۶ **DSFID-Configured-Lock-Failed**: **DSFID** به درستی پیکربندی و کدبندی شد، اما تابع قفل نمی‌تواند کامل شود.

۷ **Object-Locked-Could-Not-Modify**: کدبندی موجود بر روی برچسب RFID قفل می‌شود و در نتیجه این تلاش که مقدار **Object** را به‌روزرسانی می‌کند نمی‌تواند کامل شود.

۸ **Simulation-Id-Not-Found**: برچسب RFID ویژه، که با **Singulation-Id** مشخص شده است نمی‌تواند در ناحیه عملیاتی مشاهده شود.

۹ **Object-Not-Added**: مجموعه داده‌های **Object**, Precursor و **Object-Identifier** را نمی‌توان به Logical Memory Map اضافه کرد (مثلا به دلیل اینکه فضای حافظه ناکافی بوده است). همچنین این پاسخ زمانی به کاربرده می‌شود که برچسب RFID از یک تابع قفل پشتیبانی کند که موفق نشده است.

۱۰ **Duplicate-Object**: یک **Object-Identifier** با مقدار همانند که باید پردازش شود، پیدا شده است (اضافه شده، اصلاح شده یا حذف شده). این فرآیند صرف‌نظر شده است.

۱۱ **Object-Added-But-Not-Locked**: مجموعه داده‌هایی که به برچسب RFID اضافه شده‌اند که از یک ویژگی قفل در حافظه پشتیبانی نکرده‌اند.

۱۲ **Object-Not-Deleted**: مجموعه داده‌های precursor، **Object-Identifier** و **Object** را نمی‌توان از Logical Memory Map حذف کرد (مثلا به دلیل اینکه تابع حذف توسط برچسب RFID ویژه پشتیبانی نمی‌شود).

۱۳ **Object-Identifier-Not-Found**: فرآیند مورد انتظار را نمی‌توان اجرا کرد زیرا **Object-Identifier** تعریف شده در واقع روی برچسب RFID کدبندی نمی‌شود.

۱۴ **Object-Locked-Could-Not-Delete**: **Object** قفل شده است و بنابراین نمی‌تواند حذف شود.

۱۵ **Object-Not-Read**: نیت و مقصود برای خواندن Object تعیین شده ناموفق بوده است.

۱۶ **Objects-Not-Read**: نیت و مقصود برای خواندن یک یا چند **Objects** تعیین شده ناموفق بوده است.

۱۷ **Blocks-Locked**: عمل مورد انتظار برای پاک کردن بایت‌های کدبندی شده از یک یا چند بلوک را نمی‌توان به دست آورد زیرا بلوک‌ها از قبل قفل شده‌اند.

۱۸ **Erase-Incomplete**: عمل مورد انتظار برای پاک کردن بایت‌های کدبندی شده از یک یا چند بلوک دچار وقفه شده است و تمام بلوک‌ها پردازش نشده‌اند. این فرمان را می‌توان بار دیگر درخواست کرد تا فرآیند کامل شود.

۱۹ **Read-Incomplete**: نیت و مقصود برای خواندن محتوی کامل Logical Memory Map ناموفق بوده است زیرا تمام بلوک‌ها از برچسب RFID به Logical Memory منتقل نشده‌اند.

۲۰ **System-Info-Not-Read**: قصد و نیت برای خواندن اطلاعات سامانه ناموفق بوده است.

۲۱ **Object-Not-Modified**: مجموعه داده‌های Precursor، **Object-Identifier**، **Object** را نمی‌توان روی Logical Memory Map اصلاح کرد (مثلا، زیرا تابع اصلاح با برچسب RFID ویژه پشتیبانی نمی‌شود).

۲۲ **Object-Modified-But-Not-Locked**: **Object** در یک برچسب RFID اصلاح شده است که یک مشخصه قفل در حافظه پشتیبانی نمی‌کند.

۲۳ **Failed-To-Read-Minimum-Number-of-Tags**: حداقل تعداد برچسب‌های RFID با معیار انتخاب ویژه شناسایی نشدند، که به‌طور احتمال به دلیل یک وقفه می‌باشد.

۲۴ **Failed-To-Read-Exact-Number-of-Tags**: تعداد دقیق برچسب‌های RFID با معیار انتخاب ویژه شناسایی نشدند و به‌طور احتمال به دلیل یک وقفه می‌باشد.

۲۵ **Password-Mismatch**: **Password** ارائه شده در فرمان مطابق با کلمه عبور معادل بر روی برچسب RFID نبوده است. بنابراین عمل مربوطه صرف‌نظر شده است.

۲۶ **Password-Not-Written**: **Password** نوشته نشده است زیرا موقعیت حافظه کدبندی شده بود و یا این ویژگی پشتیبانی نشده بود.

۲۷ **Zero-Kill-Password-Error**: عملیات kill اجرا نشده است زیرا **Kill-Password** مقدار صفر دارد.

۲۸ **Kill-Failed**: قصد و نیت برای ترجمه برچسب ناموفق بوده است (مثلا به دلیل برق ناکافی).

۲۹ **Object-Not-Editable**: **Object**، که به عنوان بخشی از **Packed-Object** کدبندی شده است اصلاح نشده است زیرا قابل ویرایش نمی‌باشد.

۳۰ **Directory-Already-Defined**: یک **Packed-Object** یک نوع فهرستی تعریف شده است.

۳۱ **Packed-Object-ID-Table-Not-Recognized**: ID-Table که برای فرمان درخواست شده است توسط کدگذار یا تحقیق‌کننده پشتیبانی نشده است و کدبندی غیرممکن است که به دست آید.

۳۲ **Tag-Data-Profile-ID-Table-Not-Recognized**: ID-Table که در فرمان درخواست شده است توسط کدگذار یا تحقیق‌کننده پشتیبانی نشده است و کدبندی غیرممکن است که به دست آید.

۳۳ **Insufficient-Tag-Memory**: این عملیات ناموفق بوده است زیرا حافظه بر روی برچسب ناکافی بوده است تا مطابق با عملیات درخواست شده باشد.

۳۴ **AFI-Not-For-Monomorphic-UII**: عملیات ناموفق بوده است، زیرا AFI که در فرمان درخواست شده است برای **Monomorphic-UIIs** ثبت نمی‌شود.

۳۵ **Monomorphic-UII-OID-Mismatch**: **Object-Identifier** که در فرمان تعریف شده است مطابق با Data Constructs از استاندارد ISO/IEC 15961-2نمی‌باشد که برای این AFI، ثبت می‌کند.

۳۶ **Command-Cannot-Process-Monomorphic-UII**: یک **Monomorphic-UII** که به عنوان **Object** برای فرمانی ارائه شده است که از این نوع UII پشتیبانی نمی‌کنند.

۳۷ **Data-CRC-Not-Applied**: حداقل یکی از **Data-CRCs** درخواست شده در **Extended-DSFIS**، در داده‌ها به کاربرده نشده است.

۳۸ **Length-Not-Encoded-In-DSFID**: حداقل یکی از طول‌های درخواست شده در **Extended-DSFIS**، کدبندی نشده است.

۳۹ **Multiple-Records-Header-Not-Configured**: بعضی از فرآیندهای لازم برای پیکربندی **Multiple-Records-Header** را نمی‌توان با Data Processor پردازش کرد، که منجر به یک

رکورد ناکامل می‌شود.

۴۰ **Multiple-Records-Header-Not-Locked**: **Multiple-Records-Header** به یک برچسب RFID اضافه شده است که از یک ویژگی قفل انتخابی در حافظه پشتیبانی نکرده است.

۴۱ **File-support-Indicators-Not-Configured**: بعضی از فرآیندهای لازم برای پیکربندی **File-Support-Indicators** را نمی‌توان با Data Processor پردازش کرد، که منجر به یک کدبندی ناکامل می‌شود.

۴۲ **File-Support-Indicators-Not-Locked**: فیلد **File-Support-Indicators** به یک برچسب RFID اضافه شده است که از یک ویژگی قفل انتخابی در حافظه پشتیبانی نکرده است.

۴۳ **Data-Format-Not-Compatible-Multiple-Records-header**: **Data-Format** در فرمان و واریانسی با قواعد کنترل کننده **Data-Format** می‌باشد که در MR-header تعریف شده است. این رکورد کدبندی نشده است.

۴۴ **Access-Method-Compatible-Multiple-Records-Header**: **Access-Method** در فرمان که با قواعد **Access-Method** کنترل کننده در MR-header ناسازگارمی‌باشد. این رکورد کدبندی نشده است.

۴۵ **Sector-Identifier-Not-Compatible-Multiple-Records-Header**: شناسه بخش در **Object-Identifier** در فرمان که با قواعد شناسه بخش کنترل کننده در MR-header ناسازگار می‌باشد. این رکورد کدبندی نشده است.

۴۶ **Record-Preamble-Not-Configured**: بعضی از فرآیندهای لازم برای پیکربندی مقدمه رکورد را نمی‌توان با Data Processor پردازش کرد، که منجر به یک کدبندی ناکامل می‌شود.

۴۷ **Record-Preamble-Not-Locked**: مقدمه رکورد که به یک برچسب RFID اضافه شده است که از یک ویژگی قفل انتخابی در حافظه، پشتیبانی نکرده است.

۴۸ **Multiple-Records-Directory-Not-Present**: این فرمان برای خواندن این فهرست نمی‌تواند فراخوانی شود، زیرا فهرستی وجود ندارد.

۴۹ **Record-Not-Deleted-Preamble-Locked**: این فرمان را نمی‌توان صدا زد، به دلیل اینکه مقدمه رکورد قفل است.

۵۰ **Record-Not-Deleted-Directory-Locked**: این فرمان را نمی‌توان صدا زد، زیرا ورودی رکورد در فهرست قفل است.

۵۱ **Record-Not-Deleted-Lower-Level-Preamble-Locked**: این فرمان را نمی‌توان صدا زد، زیرا مقدمه یک رکورد سطح- پایین‌تر قفل می‌باشد.

۵۲ **Record-Not-Deleted-Encoding-Locked**: این فرمان را نمی‌توان درخواست کرد زیرا بخشی از رکورد قفل است.

۲۵۳ ISO/IEC 15961-5 Result-Code: این **Completion-Code** استفاده می‌شود تا کد Result از یک پاسخ استاندارد ISO/IEC 15961-5 به شکل {کد پیام} {مقدار نتیجه} را ببینند.

۲۵۴ **Undefined-Command-Error**: یک خطا در یک فرمان اتفاق افتاده که با **Completion-Code** دیگر تعریف نشده است.

۲۵۵ **Execution-Error**: یک خطای سامانه روی داده است که عمل فرمان را غیرممکن ساخته است. **Execution-Code** مناسب در پاسخ بر می‌گردد.

۹-۳ Execution-Code

Execution-Code بخشی از پاسخ به هر فرمان است. **Execution-Code** یک مقدار INTEGER است که مسیری را گزارش می‌کند که فرمان به وسیله سامانه با موفقیت یا بدون موفقیت پردازش و اجرا شده است. در هر پاسخ بر گردانده می‌شود. اگر مقدار آن (00_{16}) 0 باشد، این فرمان با موفقیت پردازش شده است، یعنی پروتکل اجرا شده است. مقادیر دیگر نشان می‌دهند که یک خطای سامانه روی داده است.

Execution-Codeهای زیر به کار برده می‌شوند:

۰ **No-Error**: این فرمان بدون خطا اجرا شده است.

۱ **No-Response-From-Tag**: هیچ پاسخی از برچسب RFID دریافت نشده است.

۲ **Tag-Communication-Error**: پاسخ(ها) از برچسب(های) RFID خراب شده‌اند (مثلا یک قاب قطع شده).

۳ **Tag-CRC-Error**: یک خطای CRC در پاسخ برچسب RFID کشف شده است.

۴ **Command-Not-Supported**: کد فرمان توسط تحقیق‌کننده و یا برچسب RFID، پشتیبانی نمی‌شود.

۵ **Invalid-Parameter**: پارامترهای فرمان نامعتبر هستند.

۶ **Interrogator-Communication-Error**: یک خطا در ارتباطات بین برنامه کاربردی و در تحقیق‌کننده روی داده است.

۷ **Internal-Error**: یک خطا در نرم‌افزار برنامه کاربردی روی داده است.

۲۵۵ **Undefined-Error**: یک خطا روی داده است، با کد دیگر تعریف نشده است.

۱۰ فرمان‌ها و پاسخ‌ها

هر جفت فرمان و پاسخ در زیربندهای ۱۰-۱ تا ۱۰-۲۱ تعیین می‌شود. آن متغیرهایی که به یک مشخصه جداگانه نیاز دارند، در زیربند ۱۰-۲۷-۲ تعریف می‌شوند.

۱۰-۱ Configure-AFI

۱۰-۱-۱ فرمان Configure-AFI

AFI یک کد تک بایتی است و به عنوان بخشی از فرآیند انتخابی در یک برنامه کاربردی استفاده می‌شود. جزئیات کدهای **AFI** تعیین شده در برنامه‌های کاربردی خاصی بر روی ثبت ساختارهای داده در دسترس هستند که توسط مرجع ثبت از استاندارد ISO/IEC 15961-2 تهیه شده‌اند. اگر واسط هوایی، ویژگی **AFI** را پشتیبانی کند، تنها برچسب‌ها با یک **AFI** خاص، برای پردازش آینده برگشت داده خواهند شد.

این فرمان برای آن پروتکل‌های واسط هوایی قابل کاربرد است که یکی از مشخصات زیر صحیح باشد:

الف. فرمان برنامه کاربردی به‌طورمستقیم به یک فرمان واسط هوایی معادل مربوط شود.

ب. فرمان برنامه کاربردی به **AFI** نیاز دارد تا در یک موقعیت حافظه خاص نوشته شود و به صورت ظاهری از داده‌های مربوط به اقلام جدا شود، اما از یک فرمان نوشتن واسط هوایی کلی استفاده می‌کند.

فرمان **Configure-AFI** متغیرهای زیر را دارد:

AFI (به زیربند ۷-۲-۲ مراجعه کنید)

AFI-Lock (به زیربند ۷-۴-۳ مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

متغیر **AFI-Lock** در مشخصه الف بالا، بدون هیچ قید و شرطی به کار برده می‌شود، اما تنها ممکن است در مشخصه ب بالا، به کاربرده شود اگر پروتکل واسط هوایی از قفل شدن یک موقعیت تک بایتی تعریف شده برای **AFI** پشتیبانی کند.

Configure-AFI command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 1

Singulation-Id: BYTE STRING (0..255)

AFI: BYTE
Possible Values:

Value	Definition
00_{16} – $0F_{16}$	As defined in ISO/IEC 15961-3
90_{16} – CE_{16}	As published by the Registration Authority of ISO/IEC 15961-2
CF_{16}	Reserved as an extension code

AFI-Lock: BOOLEAN
If set to TRUE, the interrogator shall lock the AFI

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰-۱-۲ پاسخ Configure-AFI

پاسخ **Configure-AFI** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Configure-AFI response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 1

Completion-Code: INTEGER

Possible Values:

Value	Definition
0	No-Error
1	AFI-Not-Configured
2	AFI-Not-Configured-Locked
3	AFI-Configured-Lock-Failed
8	Singulation-Id-Not-Found
255	Execution-Error

۱۰-۲ Configure-DSFID

Configure-Extended-DSFID (به زیربند ۱۰-۲۴ مراجعه کنید) هنگامی موردنیاز است که نشانگرهای دیگر ویژگی‌ها در برچسب RFID (از قبیل استفاده از یک **Data-CRC**) را کدبندی می‌کند. این فرمان ممکن است بهتر باشد تا یک **Access-Method** را با یک مقدار بیشتر از ۳ یا یک **Data-Format** با یک مقدار بیشتر از ۳۱ کدگذاری کند.

۱۰-۲-۱ فرمان Configure-DSFID

DSFID یک کد تک بایتی است که برای کاهش کدبندی **Object-Identifiers** استفاده می‌شود و قواعد کدبندی ویژه را برای دنبال کردن استاندارد ISO/IEC 15962 تعریف می‌کند.

به‌خصوص، این قواعد کدبندی در **Access-Method** به کاربرده می‌شوند. جزئیات **Data-Format** (بخشی از **DSFID**) که مخصوص برنامه‌های کاربردی خاص می‌باشد بر روی ثبت ساختارهای داده در دسترس هستند که با مرجع ثبت استاندارد ISO/IEC 15961-2 تهیه شده‌اند.

این فرمان برای آن پروتکل‌های واسط هوایی قابل کاربرد است که در آنجا یکی از مشخصات زیر صحیح می‌باشند:

الف. فرمان کاربردی به‌طورمستقیم به یک فرمان واسط هوایی معادل مربوط می‌شود.

ب. فرمان کاربردی به **SDFID** نیاز دارد تا در یک موقعیت خاص و حافظه نوشته شود که به‌طورظاهر از داده‌های مربوط به اقلام جدا شده است، اما از یک فرمان نوشتن واسط هوایی کلی استفاده می‌کند.

فرمان **Configure-DSFID** متغیرهای زیر را دارد:

SDFID-Constructs (به زیربند ۲-۱۱ مراجعه کنید)

DSFID-Lock (به 0 مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

متغیر **DSFID-Lock** در مشخصه الف بالا بدون قید و شرط به کاربرده می‌شود، اما تنها می‌تواند در مشخصه ب به کاربرد اگر پروتکل واسط هوایی از قفل شدن یک موقعیت تک بایتی تعریف شده با **DSFID** پشتیبانی کند.

Configure-DSFID command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 2

Singulation-Id: BYTE STRING (0..255)

DSFID-Constructs-list: List of <DSFID-Constructs>

DSFID-Lock: BOOLEAN
If set to TRUE, the interrogator shall lock the DSFID

۱۰-۲-۲ پاسخ Configure-DSFID

پاسخ **Configure-DSFID** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Configure-DSFID response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 2

Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
4	DSFID-Not-Configured
5	DSFID-Not-Configured-Locked
6	DSFID-Configured-Lock-Failed
8	Singulation-Id-Not-Found
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰-۳ Inventory-Tags

۱۰-۳-۱ فرمان Inventory-Tags

فرمان **Inventory-Tags** به تعیین مقدار **AFI** نیاز دارد تا برچسب‌های RFID متعلق به یک کلاس خاص را انتخاب کند، بطور نمونه شامل برچسب‌هایی است که به یک قلمرو معینی تعلق دارند. فرمان **Inventory-Tags** مجموعه‌ای از **Singulation-Ids** را از برچسب‌های RFID می‌خواند که یک **AFI** ویژه‌ای دارند. تنها زمانی قابل کاربرد است که یک فرمان واسط هوایی با استفاده از **AFI** به عنوان یک متغیر مشخص، از یک فرآیند فهرست پشتیبانی می‌کند و بطور کلی یک شناسه تراشه انحصاری در فرآیند داوری استفاده می‌شود.

یک معیار انتخابی دیگر (**Identify-Method**) تعیین می‌کند که چه تعداد از برچسب‌های RFID، مطابق با معیار انتخابی **AFI** مخصوص، باید قبل از تهیه پاسخ شناسایی شود. یک سازوکاری که می‌توان استفاده تا هر RFID واردشونده به ناحیه عملیاتی را کشف کند، این است که مقدار ۱ را فیلد **Inventory-At-Lest** قرار دهد. شرایط خاص را می‌توان تنها با تقبل کردن یک فهرست جزئی اثبات کرد، یعنی با استفاده از **Inventory-At-Least** یا **Inventory-No-More-Than**. یک تطبیق یک کمیت معلوم از تراکنش‌ها قبلی (مثلا، تشخیص اینکه تمام عنوان‌هایی که در یک ظرف قرار می‌گیرند در واقع آنجا هستند) را می‌توان با استفاده از فیلد **Inventory-Exactly** به دست آورد.

فرمان **Inventory-Tags** متغیرهای زیر را دارد:

AFI (به زیربند ۷-۲-۲ مراجعه کنید)

Inventory-Method (به زیربند ۷-۴-۲۳ مراجعه کنید)

Number-of-Tags (به زیربند ۷-۴-۴۳ مراجعه کنید)

Inventory-Tags command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 3

AFI: BYTE

Identify-Method: INTEGER (0..15)

Possible Values:

Value	Definition
0	Inventory-All-Tags
1	Inventory-At-Least
2	Inventory-No-More-Than
3	Inventory-Exactly
4 – 15	Reserved

Number-Of-Tags: INTEGER (0..65535)

اگر **Identify-Method** به **Inventory-All-Tags** تنظیم شود، تحقیق‌کننده باید یک فهرست کامل از تمام برچسب‌های RFID موجود در رشته عملیات خود را اجرا کند.
مقدار **Number-of-Tags** نامربوط است و باید توسط برنامه کاربردی به صفر تنظیم شود.
اگر **Identify-Method** به **Inventory-At-Least** تنظیم شود ، تحقیق‌کننده باید یک فهرست از برچسب‌های RFID موجود در رشته عملیات خود را اجرا کند و (شاید) همچنان منتظر بماند تا اینکه چند برچسب را معادل با **Number-of-Tags** مشخص کند. اگر **Number-of-Tags** به۱ تنظیم شود، تحقیق‌کننده منتظر می‌ماند تا اینکه اولین برچسب RFID کشف شود. این یک سازوکار برای منتظر بودن یک برچسب می‌باشد تا وارد فیلد تحقیق‌کننده شود. اگر **Number-of-Tags** به بیش از ۱ تنظیم شود، تحقیق‌کننده منتظر می‌ماند تا اینکه تعداد معینی از برچسب‌های RFID کشف شوند.
اگر **Identify-Method** به **Inventory-No-More-Than** تنظیم شود، تحقیق‌کننده باید یک فهرستی از برچسب‌های RFID موجود را در رشته عملیاتی خود راه‌اندازی کند و باید یک پاسخ را با چند برچسب پایین‌تر یا برابر با **Number-of-Tags** برگرداند. تحقیق‌کننده ممکن است فرآیند فهرست را متوقف کند زمانی که **Number-of-Tags** به دست آمده است یا ممکن است فرآیند فهرست ادامه داشته باشد تا اینکه تمام برچسب‌ها خوانده شوند.

یادآوری– این ممکن است توسط واسط هوایی و سازوکار ضدتصادم محدود شود.

اگر **Identify-Method** به **Inventory-Exactly** تنظیم شود، تحقیق‌کننده باید یک فهرست از برچسب‌های RFID موجود در رشته عملیات خود را راه‌اندازی کند و باید یک پاسخ را با تعدادی برچسب‌های برابر **Number-of-Tags** برگرداند. این پارامتر فرمان را می‌توان استفاده کرد تا تعداد واقعی عنوان‌های برچسب زده در ظرف، اثبات شوند. تحقیق‌کننده منتظر می‌ماند تا اینکه تعداد معینی از برچسب‌ها کشف شوند. تحقیق‌کننده ممکن است فرآیند فهرست را قطع کند زمانی که **Number-of-Tags** به دست آمده باشد و یا ممکن است به فرآیند فهرست ادامه دهد تا اینکه تمام برچسب‌ها خوانده شوند.

یادآوری– این ممکن است توسط واسط هوایی و سازوکار ضدتصادم محدود شود.

اجرای این فرمان با متغیرهای **Inventory-At-Least** و **Inventory-Exactly** می‌تواند باعث شود تا تحقیق‌کننده منتظر بماند تا اینکه برچسب‌های RFID کافی وارد رشته عملیاتی آن شود. همچنین پاسخ این فرمان را نمی‌توان تا پس از این تأخیر آغاز کرد. این مسئولیت برنامه کاربردی است که این توانایی را تطبیق دهد.

۱۰-۳-۲ پاسخ Inventory-Tags

پاسخ **Inventory-Tags** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Identities (به زیربند ۷-۵-۲ مراجعه کنید)

Number-of-Tags-Found (به زیربند ۷-۵-۹ مراجعه کنید)

Inventory-Tags response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 3

Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
23	Failed-To-Read-Minimum-Number-Of-Tags
24	Failed-To-Read-Exact-Number-Of-Tags
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

Number-Of-Tags-Found: INTEGER (1..65535)

Identities: List of <Singulation-Id>

۱۰-۴ Delete-Object

۱۰-۴-۱ فرمان Delete-Object

فرمان **Delete-Object** به تحقیق‌کننده آموزش می‌دهد تا یک **Object-Identifier** و **Object** و پارامترهای مربوطه آن را حذف کند. به ازای هر دستور، فقط یک برچسب RFID و فقط یک **Object-Identifier** باید برنامه‌نویسی شود تا اطمینان دهد که فرآیند حذف مؤثر است. تابع حذف باید **Object-Identifier**، **Object** مربوطه، پیشرو و مولفه‌های دیگر مجموعه داده‌ها از Logical Memory Map را حذف کند و سپس هر مجموعه داده‌ها را در موقعیت آدرس بالاتر روی برچسب RFID بازنویسی کنید. برای **No-Directory** و **Directory** **Access-Methods**، اگر کدبندی دیگری، **Object-Identifier** حذف شده را دنبال کند، Data Processor ممکن است بایت‌های حذف شده با یک **Data-Set** پوچ را جایگزین کند. این روش بطور خودکار با Data Processor فراخوانی می‌شود.

برای **Packed-Objects Access-Method**، تحقیق‌کننده باید **Object** را از داخل اولین **Packed-Object** که در **Object-Identifier** وجود دارد، پاک کند. اگر **Packed-Object** قابل ویرایش باشد، روش‌های ویرایش **Packed-Objects** باید استفاده شوند بنابراین حجم کل حافظه نباید بازنویسی شود. در غیر این‌صورت، تحقیق‌کننده باید محتوای حافظه کاربر را بازنویسی کند که بطور مناسبی با **Object-Identifier** و **Object**

خودش و پارامترهای مربوطه حذف شده، دوباره محاسبه شده است. اگر هر یک از حافظه‌ای که لازم است در طی اجرای این فرمان بازنویسی شود قفل شود، پاسخ، **Completion-Code** مناسب را بر می‌گرداند.

اگر **Access-Method** ، **Tag-Data-Profile** باشد، حذف امکان‌پذیر نمی‌باشد و این به دلیل ساختار ثابت کدبندی و این واقعیت می‌باشد که هیچ نویسه پوچ مبتنی بر سامانه وجود ندارد و کد تکمیل مناسب باید بر گردانده شود.

برای **Multiple-Recodes**، **Object-Identifier** باید یک اقلام داده را در یک رکورد فردی شناسایی کند. قواعد برای **Access-Method** اصلی، باید به کاربرده شوند.

برای حذف کل رکورد به زیربند ۱۰-۲۷ مراجعه کنید.

فرمان **Delete-Object** به تحقیق‌کننده آموزش می‌دهد تا مجموعه داده‌های تعیین شده با **Object-Identifier** خودش را از Logical Memory Map برچسب RFID حذف کند. اگر مجموعه داده‌ها قفل شود، این رویه ممکن است موفقیت‌آمیز نباشد؛ اگر این مورد باشد، پاسخ **Completion-Code** مناسب را بر می‌گرداند. اگر پرچم **Check-Duplicate** به TRUE تنظیم شود، تحقیق‌کننده باید قبل از حذف **Object** درخواست شده، که تنها یک نمونه **Object-Identifier** درخواستی وجود دارد، بازبینی کند. اگر تحقیق‌کننده کشف کند که برچسب RFID بیشتر از یک نمونه **Object-Identifier** مرجع را کدبندی می‌کند، نباید تابع **Delete-Object** را اجرا کند و باید **Completion-Code** مناسب را برگرداند.

اگر پرچم **Check-Duplicate** به FALSE تنظیم شود، تحقیق‌کننده باید اولین رخداد مجموعه داده‌های تعیین شده با Object-Identifier آن را حذف کند.

یادآوری- این یک متغیر است که بطور مؤثری هیچ حفاظتی را در مقابل **Object-Identifier** دو نسخه‌ای فراهم نمی‌کند. باید تنها زمانی استفاده شود که یک انتظار زیاد بدون دو نسخه‌ای وجود دارد.

فرمان **Delete-Object** متغیرهای زیر را دارد:

Check-Duplicate (به زیربند ۷-۴-۱۰ مراجعه کنید)

Object-Identifier (به زیربند ۷-۳-۲ مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

Delete-Object command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 5

Singulation-Id: BYTE STRING (0..255) **Object-**

Identifier: OBJECT IDENTIFIER **Check-**

Duplicate: BOOLEAN
If set is TRUE, the interrogator shall check that there is only one occurrence of the Object-Identifier

۱۰-۴-۲ پاسخ Delete-Object

پاسخ **Delete-Object** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Delete-Object response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 5

Completion-Code: INTEGER

Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
10	Duplicate-Object
12	Object-Not-Deleted
13	Object-Identifier-Not-Found
14	Object-Locked-Could-Not-Delete
37	Data-CRC-Not-Applied
38	Length-Not-Encoded-In-DSFID
255	Execution-Error

Execution-Code: INTEGER

Possible Values: As defined in 9.3.

۱۰-۵ Modify-Object

۱۰-۵-۱ فرمان Modify-Object

فرمان **Modify-Object** مقدار یک **Object** داده‌ها را تغییر می‌دهد که به یک **Object-Identifier** مربوط می‌شود که از قبل بر روی حافظه برچسب RFID کدبندی شده است.

حافظه کامل باید خوانده شود تا مطمئن شوید که **Object-Identifier**، دو نسخه‌ای نمی‌باشد. اگر این‌طور باشد فرمان متوقف می‌شود. صدازدن این فرمان به **Access-Method** بستگی دارد که برای برچسب RFID اعلان شده است. همچنین این رویه متفاوت است اگر طول کدبندی برآیند **Object** اصلاح شده متفاوت از طول اصلی باشد. هر یک از این موارد در **Access-Method** مناسب بحث می‌شود.

این فرمان نباید برای اصلاح یک **Monomorphic-UII** استفاده شود. اگر **AFI** بر روی برچسب RFID، اعلان کند که برای یک **Monomorphic-UII** ثبت می‌شود، خطای مناسب باید برگردانده شود و فرآیند کدبندی متوقف شود. فرمان صحیح برای استفاده در زیربند ۱۰-۲۳ تعریف می‌شود.

فرمان **Modify-Object** به تحقیق‌کننده آموزش می‌دهد تا سه فرآیند مربوطه را انجام دهد:

۱. Logical Memory Map کامل را از برچسب RFID بخواند.

۲. بسته کدبندی شده را شناسایی کند (مثلا **Data-Set** یا **Packed-Object** یا **Tag-Data-Profile**) که با **Object-Identifier** تعیین شده است. اگر نمونه‌های دو نسخه‌ای پیدا شوند، این فرآیند متوقف می‌شود.

۳. با بسته کدبندی شده اصلاح شده دوباره بنویسید:

- شامل ساخت دوباره Precursor برای یک **Data-Set**
- شامل هر یک از قواعد ساختاری **Packed-Object**
- شامل همه بایت‌های لت[1] برای داده‌های کوتاه‌تر در یک **Tag-Data-Profile**

اگر Object از قبل قفل شود، نمی‌تواند اصلاح شود و کد تکمیل مناسب باید برگشت داده شود. اگر **Access-Method**، **Packed-Objects** باشد و **Packed-Object** قابل ویرایش نباشد، **Object** را نمی‌توان اصلاح کرد و کد تکمیل مناسب باید برگشت داده شود.

اگر **Access-Method**، **Tag-Data-Profile** باشد، سه شرایط را می‌توان نشان داد:

- اگر داده‌های فشرده جدید هم طول باشد، سپس این داده‌ها به راحتی بازنویسی[2] می‌شود.
- اگر کوتاه‌تر باشد، با برچسب بایت‌های لت لازم نوشته می‌شود.
- اگر بلندتر باشد، یک خطا وجود دارد و داده‌ها را نمی‌توان اصلاح کرد و کد تکمیل مناسب باید برگشت داده شود.

اگر شی داده بخشی از یک رکورد چندگانه باشد (با **root-OID** اصلی از ۱/۰/۱۵۹۶۱/۴۰۱ شناسایی می‌شود) یا یک رکورد چندگانه سلسله مراتبی (با **root-OID** اصلی از ۱/۰/۱۵۹۶۱/۴۰۲ شناسایی می‌شود)، آنگاه فرآیند اصلاح یک شی داده باید همانی باشد که **Access-Method** را اعلان کرده است. یک شی داده را نمی‌توان بر روی یک فهرست اقلام داده اصلاح کرد (با **root-OID** اصلی از ۱/۰/۱۵۹۶۱/۴۰۳ شناسایی می‌شود).

فرمان **Modify-Object** متغیرهای زیر را دارد:

Compact-Parameter (به زیربند ۷-۳-۵ مراجعه کنید)

Object (به زیربند ۷-۳-۴ مراجعه کنید)

Object-Identifier (به زیربند ۷-۳-۲ مراجعه کنید)

Object-Lock (به زیربند ۷-۳-۶ مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

1 - Pad
2 - Overwritten

Modify-Object command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 6

Singulation-Id: INTEGER

(0..255) **Object-Identifier**:

OBJECT IDENTIFIER **Object**:

BYTE STRING

Compact-Parameter: INTEGER (0..15)

Possible Values:

Value	Definition (see 7.3.5 for further details)
0	Application-Defined
1	Compact
2	UTF8-Data
3	Packed-Objects
4	Tag-Data-Profile

Object-Lock: BOOLEAN
If TRUE the interrogator shall lock the relevant Data-Set

۱۰-۵-۲ پاسخ Modify-Object

پاسخ **Modify-Object** نام‌های فیلد زیر را دارد:

Completion Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Modify-Object response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 6

Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
7	Object-Locked-Could-Not-Modify
8	Singulation-Id-Not-Found
10	Duplicate-Object
13	Object-Identifier-Not-Found
21	Object-Not-Modified
22	Object-Modified-But-Not-Locked
33	Insufficient-Tag-Memory
36	Command-Cannot-Process-Monomorphic-UII
37	Data-CRC-Not-Applied
38	Length-Not-Encoded-In-DSFID
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰-۶ Read-Object-Identifiers

۱۰-۶-۱ فرمان Read-Object-Identifiers

این فرمان نباید با چند رکورد استفاده شود. در عوض، فرمان **Read-Multiple-Records** باید استفاده شود (به زیربند ۱۰-۲۶ مراجعه کنید)

فرمان **Read-Object-Identifiers** به تحقیق‌کننده آموزش می‌دهد که تمام **Object-Identifiers** را از برچسب RFID بخواند. این پودمان را می‌توان قبل از یک فرمان انتخابی بیشتر، برای خواندن یک **Object** ویژه یا شناسایی **Object-Identifiers** دو نسخه‌ای استفاده کرد،بطوری‌که یک رویه خانه‌داری[1] را می‌توان صدا زد. اگر Logical Memory Map برچسب RFID، هیچ **Object-Identifiers**ایی ذخیره نداشته باشد، یک پاسخ معتبر، یک فهرست **Object-Identifiers** خالی را بر می‌گرداند. فقط یک برچسب RFID باید به ازای هر فرمان برنامه‌نویسی شود تا مطمئن شویم که فرآیند خواندن مؤثر است.

فرمان **Read-Object-Identifiers** متغیر زیر را دارد:

Singulation-Id (به زیربند ۷-۲۱ مراجعه کنید)

1 - Housekeeping

Read-Object-Identifiers command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 8

Singulation-Id: BYTE STRING (0..255)

۱۰-۶-۲ پاسخ Read-Object-Identifiers

پاسخ **Read-Object-Identifiers** متغیرهای زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Read-OIDs-Response-List (به 0 مراجعه کنید)

Read-Object-Identifiers response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 8

Read-OIDs-Response-List: List of <Read-OIDs-Response>

Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰-۷ Read-Logical-Memory-Map

۱۰-۷-۱ فرمان Read-Logical-Memory-Map

مهمترین کارکرد این فرمان برای اهداف تشخیصی است، اما ممکن است برای کارکردهای دیگری استفاده شود که در آنجا خواندن محتوای کامل Logical Memory Map لازم است. فقط یک برچسب RFID باید به ازای هر فرمان برنامه‌نویسی شود تا مطمئن شویم که فرآیند خواندن مؤثر است.

فرمان **Read-Logical-Memory-Map** به تحقیق‌کننده آموزش می‌دهد که کل Logical Memory Map را از برچسب RFID بخواند و با این بدون هیچ رمزگشایی و تفسیر پاسخ دهد (یعنی توسط بازگشت مقادیر بایت کدبندی شده). هیچ پردازشی در Data Processor به عنوان بخشی از این فرمان خواندن روی نمی‌دهد، بنابراین هر یک از **Object-Identifiers**، **Objects**، **Compact-Parameter** و **Lock-Status** را نمی‌توان به‌طورمستقیم مشخص کرد.

این فرمان بطور یکسان در تمام **Access-Method** به کاربرده می‌شود، اما اگر یک ساختار **Directory** توسط **Access-Method** تعریف شده باشد، این باید در پاسخ قرار گیرد، اما نباید از بایت‌های دیگر در Logical Memory Map تشخیص داده شود.

فرمان **Read-Logical-Memory-Map** متغیر زیر را دارد:

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

Read-Logical-Memory-Map command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 10

Singulation-Id: BYTE STRING (0..255)

۱۰-۷-۲ پاسخ Read-Logical-Memory-Map

پاسخ **Read-Logical-Memory-Map** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Logical-Memory-Map (به زیربند ۷-۵-۶ مراجعه کنید)

Read-Logical-Memory-Map response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 10

Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
19	Read-Incomplete
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

Logical-Memory-Map: BYTESTRING

۱۰-۸ Erase-Memory

۱۰-۸-۱ فرمان Erase-Memory

فرمان **Erase-Memory** به تحقیق‌کننده آموزش می‌دهد تا کل Logical Memory Map را از برچسب RFID ویژه را به مقدار صفر مجددا تنظیم کند. اگر به عنوان **Access-Method** تعریف شود، این شامل **Directory** است. اگر هیچ‌کدام از بلوک‌ها قفل نباشد، باید منجر به حذف تمام **Data-Sets** یا **Packed-Object** شود. اگر هر بلوک قفل شود، سپس **Completion-Code-Blocks-Locked** بازگردانده خواهد شد. تنها یک برچسب RFID باید به ازای هر فرمان برنامه‌نویسی شود تا مطمئن شویم که فرآیند پاک کردن مؤثر است.

پردازش بعدی توسط برنامه کاربردی، به‌طوراحتمال با خواندن تمام داده‌های قفل شده، می‌تواند فراخوانی شود تا تعیین کنند که آیا برچسب RFID همچنان قابل‌استفاده است. برای مثال، این بلوک‌ها که قفل هستند می‌توانند شامل داده‌هایی باشند که بطور دائم به اقلام اختصاص داده می‌شوند و بلوک‌های قفل نشده شامل داده‌های انتقالی می‌باشند.

فرمان **Erase-Memory** متغیر زیر را دارد:

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

Erase-Memory command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 12

Singulation-Id: BYTE STRING (0..255)

۱۰-۸-۲ پاسخ Erase-Memory

پاسخ **Erase-Memory** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Erase-Memory (به زیربند ۹-۳ مراجعه کنید)

Erase-Memory response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 12

Completion-Code: INTEGER

Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
17	Blocks-Locked
18	Erase-Incomplete
255	Execution-Error

Execution-Code: INTEGER

Possible Values: As defined in 9.3.

۱۰-۹ Get-App-Based-System-Info

۱۰-۹-۱ فرمان Get-App-Based-System-Info

فرمان **Get-App-Based-System-Info** به تحقیق‌کننده آموزش می‌دهد تا اطلاعات سامانه را بخواند و آن متغیرهایی را برگرداند که به برنامه کاربردی مربوط هستند، یعنی **AFI** و **DSFID**. این فرمان برای انواع برچسب RFID که این کدها را به عنوان بخشی از یک پاسخ به فرمان‌های دیگر برگشت نمی‌دهد، مفید است.

فرمان **Get-App-Based-System-Info** متغیر زیر را دارد:

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

Get-App-Based-System-Info command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 13

Singulation-Id: BYTE STRING (0..255)

۱۰-۹-۲ پاسخ Get-App-Based-System-Info

پاسخ **Get-App-Based-System-Info** متغیرهای زیر را دارد:

AFI (به زیربند ۷-۲-۲مراجعه کنید)

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

DSFID (به زیربند ۷-۲-۳ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Get-App-Based-System-Info response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 13

AFI: BYTE

DSFID: BYTE STRING

Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
20	System-Info-Not-Read
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰-۱۰ Write-Objects

۱۰-۱۰-۱ فرمان Write-Objects

فرمان **Write-Objects** برای نوشتن یک یا چند **Object-Identifiers** و **Objects** مربوط به یک برچسب RFID، استفاده می‌شود. این فرمان ممکن است اجرا شود تا داده‌های اولیه را در برچسب RFID بنویسند، و یا داده‌ها را به برچسب اضافه کنند. این فرمان با یک متغیر ترکیبی **Add-Objects-List** پشتیبانی می‌شود.

این فرمان نباید برای نوشتن یک **Monomorphic-UII** استفاده شود. اگر **AFI** بر روی برچسب RFID، اعلان کند که این برای یک **Monomorphic-UII** ثبت می‌شود، خطای مناسب باید برگشت داده شود و فرآیند کدبندی متوقف شود.

فرمان صحیح برای استفاده، در زیربند ۱۰-۲۳ تعریف می‌شود.

DSFID-Constructs-List برای تعیین **Access-Method** و **Data-Format** استفاده می‌شود. **Ext-DSFID-Constructs-list** استفاده می‌شود تا نشانگرها را بر روی **Extended-DSFID** قرار دهد و به Data Processor آموزش دهد که رویه‌های خاصی را اجرا کند، مثلا، به کار بردن CRC در داده‌ها.

DSFID-Constructs-list و **Ext-DSFID-Constructs-list** برای استفاده در یکی از روش‌های زیر فراهم شده‌اند:

- اگر داده‌ها در یک برچسب RFID خالی نوشته شود، سپس آن‌ها به عنوان بخشی از این فرمان تهیه می‌شوند تا ارتباطات را به حداقل برسانند.
- اگر داده‌ها به برچسب RFID اضافه شود، سپس **DSFID** در فرمان، باید مطابق با **DSFID**ایی باشد که از قبل بر روی برچسب RFID کدبندی شده است، وگرنه یک خطا وجود دارد و فرآیند کدبندی می‌تواند قبل از اینکه مقدار زیادی از داده‌ها پردازش شوند، پاک کند.
- همچنین تمام الزامات اعلان شده با متغیرها در **Ext-DSFID-Constructs-List** باید پردازش شوند.

متغیر **DSFID-Lock**، اگر تنظیم شود، در تمام رشته بایت **DSFID** و **Extended-DSFID** به کار برده می‌شود.

اگر **Access-Method**، **Packed-Objects** باشد، تمام **Objects** تعیین شده به عنوان متغیرها باید در همان **Packed-Objects** جدید افزوده شده پس از هر **Packed-Objects** موجود در حافظه، قرار گیرند. تعدادی از متغیرها فقط در **Packed-Objects** به کاربرده می‌شوند و اینها در متغیر **Packed-Object-Constructs** تعریف می‌شوند.

اگر **Access-Method**، **Tag-Data-Profile** باشد، تمام **Objects** تعیین شده به عنوان متغیرها باید در همان **Tag-Data-Profile** قرار گیرند. باید با افزودن یک Object که با متغیر **Tag-Data-Profile-ID-Table** تعیین نمی‌شود، به عنوان یک خطا رفتار شود و هیچ کدبندی رخ ندهد.

یک **Multiple-Records** توسط **Object-Identifier** در **Add-Objects-list** اعلان می‌شود که همه آن‌ها به یکی از این سه شکل اجباری می‌باشند:

- **1.0.15961.401. {data format = dictionary}**.{sector identifier}.{record type}.{instance-of}.{Relative-OID of data element}

- **1.0.15961.402.{data format = dictionary}**.{sector identifier}.{record type}.{hierarchical id}.{ Relative-OID of data element}

- **1.0.15961.403.{data format = dictionary}**.{sector identifier}.{record type}.{hierarchical id}.{ Relative-OID of data element}.{list element number}

متغیر **Add-Objects-list** در یک رکورد تک، در یک ساختار رکوردهای چندگانه به کاربرده می‌شود. بنابراین در ساختار OID فهرست شده بالا تنها مقادیر قوس نهایی مجاز هستند که در یک فرمان واحد متفاوت باشند.

رکوردهای فردی باید تنها پس از ساخت و ایجاد MR-header نوشته شوند.

متغیر **DSFID-Constructs-List** باید استفاده شود تا اعلان کنند که Multiple-Record فردی مطابق با قواعد کدبندی یکی از **Access-Methods** زیر می‌باشد:

- No-Directory

۲ Packed-Objects

۳ Tag-Data-Profile

برای رکوردهای چندگانه، این فرمان به متغیر **Multiple-Records-Constructs-list** نیاز دارد (به زیربند ۸-۱۱ مراجعه کنید) تا تعریف شود. این شامل دستورالعمل‌هایی در مورد حفظ و نگهداری حافظه می‌باشد تا اندازه رکورد افزایش یابد و نشان دهد که آیا یک ورودی فهرست لازم می‌باشد و در بعضی موارد مشخص می‌کند که آیا یک رابطه والد- فرزند بین این رکورد و بقیه وجود دارد. همچنین یک متغیر وجود دارد که اعلان می‌کند آیا این رکورد به عنوان یک فهرست اقلام داده تعریف می‌شود که در این حالت، **Add-Objects-List** شامل چند نمونه از **Relative-OID** همانند می‌باشد. این فرمان ممکن است برای افزودن اقلام داده‌ها به یک رکورد چندگانه موجود استفاده شود که این کار توسط اعلان این متغیر مربوطه در **Multiple-Records-Constructs-list** انجام می‌شود.

همچنین، **Ext-DSFID-Constructs-List** نیز برای تعریف تمام الزامات کدبندی شده برای یک **Multiple-Record** لازم است.

MR-header قواعد را برای **Access-Method**، **Data-Format** و شناسه بخش تعریف می‌کند که بر کدبندی بعدی رکوردهای فردی نظارت می‌کنند. با توجه به زمینه‌های فردی آن‌ها، این قواعد به متغیرهایی نیاز داشته‌اند تا در تمام رکوردها همانند باشند و یا مجاز باشند که متفاوت باشند. هر مغایرت با MR-header باید مورد فرمان را متوقف کند و رکورد کدبندی نشود.

متغیر **DSFID-Lock** برای رکوردهای چندگانه مناسب نمی‌باشد زیرا این بخشی از مقدمه رکورد است، سپس باید به عنوان یک واحدی برای قفل کردن یا قفل نکردن در نظر گرفته شود.

فرمان **write-Objects** متغیرهای زیر را دارد:

Add-Objects-List (به زیربند ۱۱-۱ مراجعه کنید)

DSFID-Constructs (به زیربند ۱۱-۲ مراجعه کنید)

DSFID-Lock (به 0 مراجعه کنید)

Ext-DSFID-Constructs (به زیربند ۱۱-۴ مراجعه کنید)

Multiple-Records-Constructs (به زیربند ۱۱-۸ مراجعه کنید)

Packed-Object-Constructs (به زیربند ۱۱-۱۲ مراجعه کنید)

Tag-Data-Profile-ID-Table (به زیربند ۷-۴-۶۰ مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱مراجعه کنید)

Write-Objects command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 17

Singulation-Id: BYTE STRING (0..255)

DSFID-Constructs-list: [OPTIONAL] List of <DSFID-Constructs>

Ext-DSFID-Constructs-list: [OPTIONAL] List of <Ext-DSFID-Constructs>

DSFID-Lock: BOOLEAN
If set to TRUE, the interrogator shall lock the DSFID

Add-Objects-List: List of <Add-Objects>

Packed-Object-Constructs: [OPTIONAL] List of <Packed-Object-Constructs>

Tag-Data-Profile-ID-Table: INTEGER [OPTIONAL]

Multiple-Records-Constructs: [OPTIONAL] List of <Multiple-Records-Constructs>

۱۰-۱۰-۲ پاسخ Write-Objects

پاسخ **write-Objects** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

write-Responses (به زیربند ۱۱-۱۸ مراجعه کنید)

Completion-Codes اضافی در هر **Object-Identifier** استفاده می‌شود و در متغیر **Write-Responses** ترکیب می‌شوند.

Write-Objects response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 17

Write-Responses-List: List of <Write-Responses>

Completion-Code: INTEGER

Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
29	Object-Not-Editable
31	Packed-Object-ID-Table-Not-Recognised-No-Encoding
32	Tag-Data-Profile-ID-Table-Not-Recognised
33	Insufficient-Tag-Memory
36	Command-Cannot-Process-Monomorphic-UII
37	Data-CRC-Not-Applied
38	Length-Not-Encoded-In-DSFID
43	Data-Format-Not-Compatible-Multiple-Records-Header
44	Access-Method-Not-Compatible-Multiple-Records-Header
45	Sector-Identifier-Not-Compatible-Multiple-Records-Header
46	Record-Preamble-Not-Configured
47	Record-Preamble-Not-Locked
255	Execution-Error

Additional Completion-Codes apply to the Write-Response-List

Execution-Code: INTEGER

Possible Values: As defined in 9.3.

۱۰-۱۱ Read-Objects

۱۰-۱۱-۱ فرمان Read-Objects

این فرمان نباید برای خواندن هر شی داده یا بخش دیگری از یک رکورد چندگانه استفاده شود. رویه‌های مناسب در زیربند ۱۰-۲۶ تعریف می‌شوند.

فرمان **Read-Objects** به تحقیق‌کننده آموزش می‌دهد تا یک مجموعه‌ای از یک یا چند **Object-Identifier** و Objects مربوطه را از برچسب RFID بخواند. یک متغیر فرمان را می‌توان استفاده کرد تا بررسی شود که **Object-Identifier** بر روی برچسب RIFD دو نسخه‌ای نمی‌شود. فقط یک برچسب RFID باید به ازای هر فرمان برنامه‌نویسی شود تا تضمین کنیم که فرآیند خواندن مؤثر است.

این فرمان از متغیری پشتیبانی می‌کند که برنامه کاربردی را فعال می‌کند تا یک نقطه آدرس بالاتر را بر روی برچسب RFID فرمان دهد که دورتر از آن خواندن ادامه ندارد. این متغیر، ویژگی‌هایی فرمان اصلی **readFirstObject** را ترکیب می‌کند، اما تنها در یک **Object-Identifier** محدود نمی‌شود. این متغیر فرمان از ویژگی‌ها در واسط هوایی پشتیبانی می‌کند که می‌توانند سریع‌تر از خواندن یک عنوان **Object-Identifier(s)**

باشند. برنامه کاربردی ممکن است انتخاب کند که اغلب بیشترین دسترسی را به **Object(s)** که ابتدا در برچسب ذخیره شده‌اند، داشته باشد. برای متغیر **Max-App-Length**، یک مقدار باید تعیین شود (به شکل بایت) که با شبیه‌سازی کدبندی یا با یک دوره آزمایش کوتاه می‌تواند به دست آید، مقدار این متغیر تغییر می‌یابد تا زمانیکه فرمان، به محتمل‌ترین خواندن **Object-Identifier(s)** و **Object(s)** درخواست شده، دست یابد.

این فرمان ممکن است برای خواندن یک **Monomorphic-UII** استفاده شود که توسط اعلان **Object-Identifier** مربوطه به عنوان ورودی تک در فهرست **Read-Objects** که با 4 **Read-Type** ترکیب شده است، می‌باشد. Data Processor بررسی می‌کند که **Object-Identifier** به عنوان بخشی از یک ورودی **Monomorphic-UII** بر روی ساختارهای داده در استاندارد ISO/IEC 15961-2 ثبت می‌شود.

- اگر این‌طور باشد، بایت‌های کدبندی شده در قواعد تعریف شده بر روی ثبات فشرده می‌شوند و در پاسخ قرار می‌گیرند.
- اگر این‌طور نباشد، خطای مناسب بر گردانده می‌شود.

فرمان Read-Object متغیرهای زیر را دارد:

Max-App-Length (به زیربند ۷-۴-۳۲ مراجعه کنید)

Read-Objects (به زیربند ۱۱-۱۳ مراجعه کنید)

Read-Type (به زیربند ۷-۴-۵۳ مراجعه کنید)

Singulation- Id (به زیربند ۷-۲-۱ مراجعه کنید)

Read-Objects command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 18

Singulation-Id: BYTE STRING (0..255)

Read-Type: INTEGER

Possible Values:

Value	Definition
0	Read-1st-Objects
1	Read-Multiple-Objects
2	Read-All-Objects
3	Read-Monomorphic-UII

Max-App-Length: INTEGER (1..65535)
This only applies to Read-Type (0) and is expressed in bytes

Read-Objects-List: List of <Read-Objects> This does not apply to Read-All-Objects (2)

اگر متغیر **Check-Duplicate** در متغیر **Read-Object-list** به FALSE تنظیم شود، **Object-Identifier** درخواستی به دست می‌آید و تحقیق‌کننده باید اولین **Object** پیدا شده را بدون بررسی دو نسخه‌ای‌ها برگرداند. اگر متغیر **Check-Duplicate** به TRUE تنظیم شود، تحقیق‌کننده باید برای **Object-Identifier** دو نسخه‌ای، بررسی کند. اگر بیشتر از یک نمونه **Object-Identifier** درخواستی پیدا شود، تحقیق‌کننده باید ابتدا **Object** پیدا شده را برگرداند و وجود دو نسخه‌ای‌ها را با **Completion-Code** مناسب شناسایی کند.

۱۰-۱۱-۲ پاسخ Read-Objects

پاسخ **Read-Objects** متغیرهای زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Read-Object-Response-List (به زیربند ۱۱-۱۴ مراجعه کنید)

Read-Objects response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 18

Read-Objects-Response-List: List of <Read-Objects-Response>

Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
255	Execution-Error

Additional Completion-Codes apply to the Read-Objects-Response-List

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰-۱۲ Write-Objects-Segmented-Memory-Tag

۱۰-۱۲-۱ فرمان Write-Objects-Segmented-Memory-Tag

فرمان **Write-Objects-Segmented-Memory-Tag** شبیه به فرمان **Write-Objects** می‌باشد به جز اینکه داده‌ها را در یک بانک حافظه انتخابی در یک برچسب حافظه قطعه‌بندی شده می‌نویسد. این فرمان ممکن است پیاده‌سازی شود تا داده‌های اولیه را در برچسب RFID بنویسد و یا داده‌ها را به برچسب اضافه کند. این فرمان با یک متغیر ترکیبی **Add-Objects-List** پشتیبانی می‌شود.

این فرمان نباید برای نوشتن یک **Monomorphic-UII** استفاده شود. اگر AFI بر روی بر روی برچسب RFID اعلان کند که این برای یک **Monomorphic-UII** ثبت می‌شود، خطای مناسب باید برگردانده شود و فرآیند کدبندی قطع شود. فرمان صحیح برای استفاده در زیربند۱۰-۲۳ تعریف می‌شود.

Access-Password در این فرمان، استفاده می‌شود تا آن را برروی برچسب RFID تطبیق دهد و به نوشتن داده‌ها در برچسب RFID اجازه دهد.

DSFID-Constructs-List برای تعیین **Access-Method** و **Data-Format** استفاده می‌شود. **Ext-DSFID-Constructs-List** استفاده می‌شود تا نشانگرها را بر روی **Extended-DSFID** قرار دهد و به Data Processor آموزش دهند که رویه‌های خاصی را انجام دهد، مثلا، استفاده از یک CRC در داده‌ها.

متغیر **DSFID-Lock**، اگر تعیین شود، در تمام رشته بایت **DSFID** و **Extended-DSFID** به کار برده می‌شود.

AFI و **DSFID** به عنوان متغیرهایی برای استفاده در یکی از روش‌های زیر فراهم شده‌اند:

- اگر داده‌ها در یک برچسب RFID حافظه قطعه‌بندی شده خالی نوشته شود، سپس این متغیرها به عنوان بخشی از این فرمان تهیه می‌شوند تا ارتباطات را به حداقل برسانند.
- با توجه به بانک حافظه مورد نظر، اگر داده‌ها به برچسب RFID حافظه قطعه‌بندی شده اضافه شود، سپس **AFI** و **DSFID** در فرمان باید مطابق با **AFI** و **DSFID** باشند که از قبل در برچسب RFID کدبندی شده‌اند، وگرنه یک خطا وجود دارد و فرآیند کدبندی می‌تواند پاک کند قبل از اینکه مقدار زیادی از داده‌ها پردازش شوند.
- همچنین تمام شرایط اعلان شده با متغیرها در **Ext-DSFID-Constructs-List** باید پردازش شوند.

اگر **Access-Method**، **Packed-Objects** باشد، تمام **Objects** تعیین شده به عنوان متغیرها باید در همان **Packed-Objects** جدید اضافه شده، بعد از همه **Packed-Objects** موجود در حافظه، قرار گیرند. چندین متغیر، فقط در **Packed-Objects** به کاربرده می‌شود و اینها در متغیر **Packed-Object-Constructs** تعریف می‌شوند.

اگر **Access-Method**، **Tag-Data-Profile** باشد، تمام **Objects** تعیین شده به عنوان متغیرها، باید در همان **Tag-Data-Profile** گنجانده شوند. افزودن یک **Object** که با متغیر **Tag-Data-Profile-ID-Table** تعیین نمی‌شود باید به عنوان یک خطا بحث شود و هیچ کدبندی روی نمی‌دهد.

چند رکورد ممکن است تنها برروی Memory Bank 11 کدبندی شوند. رکوردهای فردی تنها پس از ایجاد MR-header نوشته می‌شوند. جزئیات تعریف شده در ۱۰/۱۰/۱ برای Multiple Records، باید در ساخت فرمان به کاربرده شود.

فرمان **write-Objects-Segmented-Memory-Tag** متغیرهای زیر را دارد:

Access-Password (به زیربند ۷-۴-۱ مراجعه کنید)

Add-Objects-list (به زیربند ۱۱-۱ مراجعه کنید)

AFI (به زیربند ۷-۲-۲ مراجعه کنید)

DSFID-Constructs (به زیربند ۱۱-۲ مراجعه کنید)

DSFID-Lock (به 0 مراجعه کنید)

Ext-DSFID-Constructs (به زیربند ۱۱-۴ مراجعه کنید)

Memory-Bank (به زیربند ۷-۴-۳۳ مراجعه کنید)

Multiple-Records-Constructs (به زیربند ۱۱-۸ مراجعه کنید)

Packed-Objects-Constructs (به زیربند ۱۱-۱۲ مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

Tag-Data-Profile-ID-Table (به زیربند ۷-۴-۶۰ مراجعه کنید)

Write-Objects-Segmented-Memory-Tag command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 19

Singulation-Id: BYTE STRING (0.255)

Memory-Bank: BIT STRING

Possible Values:

Value	Definition
01	UII-Memory
11	User-Memory

Access-Password: [Optional] BYTE STRING (4 bytes)

AFI: BYTE [only applies if Memory-Bank = 01]

DSFID-Constructs-list: List of <DSFID-Constructs>

Ext-DSFID-Constructs-list: [Optional] List of <Ext-DSFID-Constructs>

DSFID-Lock: BOOLEAN
If set to TRUE, the interrogator shall lock the DSFID

Add-Objects-List: List of <Add-Objects>

Packed-Object-Constructs: [OPTIONAL] List of <Packed-Object-Constructs>

Tag-Data-Profile-ID-Table: INTEGER [OPTIONAL]

Multiple-Records-Constructs: [OPTIONAL] List of <Multiple-Records-Constructs>

۱۰-۱۲-۲ پاسخ Write-Objects-Segmented-Memory-Tag

پاسخ **Write-Objects-Segmented-Memory-Tag** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Write-Responses (به زیربند ۱۱-۱۸ مراجعه کنید)

Completion-Codes اضافی در هر **Object-Identifier** به کاربرده می‌شوند و در متغیر **Write-Response** ترکیب می‌شوند.

Write-Objects-Segmented-Memory-Tag response

Module-OID: OBJECT IDENTIFER = 1 0 15961 127 19

Write-Response-List: List of <Write-Responses>

Completion-Code: INTEGER

Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
25	Password-Mismatch
26	AFI-Mismatch
27	DSFID-Mismatch
33	Insufficient-Tag-Memory
36	Command-Cannot-Process-Monomorphic-UII
37	Data-CRC-Not-Applied
38	Length-Not-Encoded-In-DSFID
43	Data-Format-Not-Compatible-Multiple-Records-Header
44	Access-Method-Not-Compatible-Multiple-Records-Header
45	Sector-Identifier-Not-Compatible-Multiple-Records-Header
46	Record-Preamble-Not-Configured
47	Record-Preamble-Not-Locked
255	Execution Error

Additional Completion-Codes apply to the Write-Response-List

Execution-Code: INTEGER

Possible Values: As defined in 9.3.

۱۰-۱۳ Write-EPC-UII

۱۰-۱۳-۱ فرمان Write-EPC-UII

فرمان **Write-EPC-UII** به تحقیق‌کننده آموزش می‌دهد که یک **EPC-Code** را در Memory Bank 01 از یک برچسب RFID حافظه قطعه‌بندی شده بنویسد. این فرمان از طول‌های مختلف **EPC-Codes** پشتیبانی می‌کند که با EPC global تعیین شده است.

Access-Password در این فرمان، استفاده می‌شود تا آن را برروی برچسب RFID انطباق دهد و در مقابل نوشتن غیرمجاز داده‌ها در برچسب RFID محافظت کند.

این فرمان را می‌توان استفاده کرد تا در ابتدا **EPC-Code** را در برچسب RFID بنویسد و یا مقدار کد را دوباره‌نویسی کند. اگر کد جدید، طول کوتاه تری دارد، تحقیق‌کننده باید مطمئن شود که بایت‌هایی که بخشی از کد قدیمی‌تر را نشان می‌دهند، حذف می‌شوند.

فرمان **Write-EPC-UII-Tag** متغیرهای زیر را دارد:

Access-Password (به زیربند ۷-۴-۱ مراجعه کنید)

EPC-Code (به زیربند ۷-۴-۱۹ مراجعه کنید)

Memory-Bank-Lock (به زیربند ۷-۴-۳۴ مراجعه کنید)

NSI-Bits (به زیربند ۷-۴-۴۰ مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

Write-EPC-UII command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 20

Singulation-Id: BYTE STRING (0.255)

Access-Password: [Optional] BYTE STRING (4 bytes)

NSI-Bits: BIT STRING

EPC-Code: BYTE STRING

Memory-Bank-Lock: BOOLEAN
If TRUE the entire memory bank shall be locked.

۱۰-۱۳-۲ پاسخ Write-EPC-UII

پاسخ **Write-EPC-UII** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Write-EPC-UII response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 20

Completion-Code:
INTEGER
Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
25	Password-Mismatch
33	Insufficient-Tag-Memory
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰-۱۴ Inventory-ISO-UIImemory

۱۰-۱۴-۱ فرمان Inventory-ISO-UIImemory

فرمان **Inventory-ISO-UIImemory** محتوای حافظه UII را از چند برچسب حافظه قطعه‌بندی شده بر می‌گرداند و این انتظار را دارد که یک **Object-Identifier** برای یک UII، متفاوت از یک **EPC-Code**، کدبندی شود. این پاسخ، محتوای حافظه UII را برای تمام برچسب‌ها بر می‌گرداند که رشته بیت کدبندی شده آن مطابق با متغیرهای فرمان می‌باشد.

متغیرهای تهیه شده در فرمان، یک پوشش بیت[1] را فعال می‌کنند تا در فرمان‌های پروتکل واسطه هوایی مناسب ترکیب شود تا تنها برچسب‌هایی را انتخاب کنند که مطابق با پوشش بیت هستند. **AFI** یک متغیر لازم است و دو متغیر دیگر، رشته بیت را وسعت می‌دهند تا قابلیت یک انتخاب را افزایش دهند.

این فرمان ممکن است برای خواندن یک **Monomorphic-UII** استفاده شود که **AFI** مربوط را به عنوان یک متغیر اعلان می‌کند. Data Processor بررسی می‌کند که AFI به عنوان بخشی از یک ورودی **Monomorphic-UII** روی ثبات Data Constructs از استاندارد ISO/IEC 15961-2 می‌باشد.

- اگر این‌طور باشد، بایت‌های کدبندی شده در قواعدی فشرده می‌شوند که بر روی ثبات تعریف شده‌اند و در پاسخ گنجانده شده‌اند.
- اگر این‌طور نباشد، خطای مناسب بر گردانده می‌شود.

یادآوری- زمانی که این فرمان در **Monomorphic-UII** فهرست فراخوانی می‌شود، **DSFID** لازم نیست.

فرمان **Inventory-Iso-UII memory** متغیرهای زیر را دارد:

1 - Mask

Additional-App-Bits (به زیربند ۷-۴-۲ مراجعه کنید)

AFI (به زیربند ۷-۲-۲ مراجعه کنید)

DSFID-Constructs (به زیربند ۱۱-۲ مراجعه کنید)

Inventory-ISO-UIImemory command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 21

AFI: BYTE

DSFID-Constructs-list: [OPTIONAL] List of <DSFID-Constructs>

Additional-App-Bits: BIT STRING [Optional]

۱۰-۱۴-۲ پاسخ Inventory-ISO-UIImemory

پاسخ **Inventory-ISO-UIImemory** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

ISO-UIImemory (به زیربند ۱۱-۵ مراجعه کنید)

Inventory-ISO-UIImemory response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 21

ISO-UII Memory-List: List of <ISO-UIIMemory>

Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰-۱۵ Inventory-EPC-UIImemory

۱۰-۱۵-۱ فرمان Inventory-EPC-UIImemory

فرمان **Inventory-EPC-UIImemory** محتوای حافظه UII را از چند برچسب حافظه قطعه‌بندی شده بر می‌گرداند، با این انتظار که **EPC-Code** کدبندی می‌شود. این پاسخ، محتوای حافظه UII را برای تمام برچسب‌هایی بر می‌گرداند که رشته بیت کدبندی شده آن مطابق با متغیرهای فرمان می‌باشد.

متغیرهای تهیه شده در فرمان، یک پوشش بیت را فعال می‌کنند تا در فرمان‌های پروتکل واسط هوایی مناسب ترکیب شود تا تنها برچسب‌هایی را انتخاب کنند که مطابق با پوشش بیت می‌باشد. مقدار **Tag-Mask** شامل یک رشته بیت است که باید با رجوع به استانداردهای EPC global مناسب، تعیین شود و در **Pointer**ایی به کاربرده می‌شودکه اولین بیت یک رشته پیوسته را بر روی برچسب RFID که باید مطابق با Tag-Mask باشد، شناسایی می‌کند.

فرمان **Inventory-EPC-UII memory** متغیرهای زیر را دارد:

Length-Of-Mask (به زیربند ۷-۴-۲۷ مراجعه کنید)

Pointer (به زیربند ۷-۴-۵۰ مراجعه کنید)

Tag-Mask (به زیربند ۷-۴-۶۱ مراجعه کنید)

Inventory-EPC-UIImemory command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 22

Pointer: HEXADECIMAL ADDRESS
The address of the first (msb) bit against which to apply the Tag-Mask.

Length-Of-Mask: INTEGER

Tag-Mask: BIT STRING

۱۰-۱۵-۲ پاسخ Inventory-EPC-UIImemory

پاسخ **Inventory-EPC-UII memory** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

EPC-UII memory (به زیربند ۱۱-۳ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Inventory-EPC-UIImemory response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 22

EPC-UIImemory-List: List of <EPC-UIImemory>

Completion Code: INTEGER

Possible Values:

Value	Definition
0	No-Error
255	Execution-Error

Execution-Code: INTEGER

Possible Values: As defined in 9.3.

۱۰-۱۶ Write-Password-Segmented-Memory-Tag

۱۰-۱۶-۱ فرمان Write-Password-Segmented-Memory-Tag

فرمان **Write-Password-Segmented-Memory-Tag** به تحقیق‌کننده آموزش می‌دهد که یکی از **Passwords** تعریف شده در فرمان را در حافظه مناسبی از یک برچسب RFID حافظه قطعه‌بندی شده بنویسد. به ازای هر فرمان، فقط یک **Password** ممکن است تعیین شود. **Password-Type**، به Data Processor، نوع **Password** را می‌شناساند و بنابراین موقعیت ذخیره‌سازی آن روی برچسب RFID را مشخص می‌کند.

فرمان **Write-Password-Segmented-Memory-Tag** متغیرهای زیر را دارد:

Password (به زیربند ۷-۴-۴۶ مراجعه کنید)

Password-Type (به زیربند ۷-۴-۴۷ مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

Write-Password-Segmented-Memory-Tag command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 23

Singulation-Id: BYTE STRING (0..255)

Password-Type: INTEGER

Possible Values:

Value	Definition
0	Kill-Password
1	Access-Password

Password: BYTE STRING

For **Password-Types** 0 and 1, the length is 4 bytes

۱۰-۱۶-۲ پاسخ Write-Password-Segmented-Memory-Tag

پاسخ **write-Password-Segmented-Memory-Tag** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Write-Password-Segmented-Memory-Tagresponse

Module-OID: OBJECT IDENTIFIER = 1 – 15961 127 23

Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
26	Password-Not-Written
33	Insufficient-Tag-Memory
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰-۱۷ Read-Words-Segmented-Memory-Tag

۱۰-۱۷-۱ فرمان Read-Words-Segmented-Memory-Tag

فرمان **Read-Words-Segmented-Memory-Tag** به تحقیق‌کننده آموزش می دهد که یک ترتیب پیوسته‌ای از کلمات را از یکی از بانک‌های حافظه یک برچسب RFID حافظه قطعه‌بندی شده بخواند. این فرمان را می‌توان استفاده کرد تا بایت‌های کدبندی شده استخراج شوند، که ممکن نیست بر مبنای **Object** باشند از قبیل **Singulation-Id** انحصاری یا یک کلمه عبور. همچنین، ممکن است برای اهداف تشخیص مفید باشد.

فرمان **Read-Words-Segmented-Memory-Tag** متغیرهای زیر را دارد:

Access-Password (به زیربند ۷-۴-۱ مراجعه کنید)

Memory-Bank (به زیربند ۷-۴-۳۳ مراجعه کنید)

Singulation-Id (به زیربند ۷-۴-۶۳ مراجعه کنید)

Word-Pointer (به زیربند ۷-۴-۶۴ مراجعه کنید)

Read-Words-Segmented-Memory-Tag command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 24

Singulation-Id: BYTE STRING (0.255)

Memory-Bank: BIT STRING
Possible Values: (00..11)

Word-Pointer: HEXADECIMAL ADDRESS

Word-Count: INTEGER

Access-Password: [Optional] BYTE STRING (4 bytes)

۱۰-۱۷-۲ پاسخ Read-Words-Segmented-Memory-Tag

پاسخ **Read-Words-Segmented-Memory-Tag** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Read-Data (به زیربند ۷-۵-۱۲ مراجعه کنید)

Read-Words-Segmented-Memory-Tag response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 24

Read-Data: BYTE STRING

Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
25	Password-Mismatch
254	Undefined-Command-Error
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰-۱۸ Kill-Segmented-Memory-Tag

۱۰-۱۸-۱ فرمان Kill-Segmented-Memory-Tag

فرمان **Kill-Segmented-Memory-Tag** به تحقیق‌کننده آموزش می‌دهد تا از پروتکل‌های واسط هوایی مناسب استفاده کند تا برچسب RFID غیرقابل خواندن را در آینده ارائه دهد. **Kill-Password** در این فرمان باید مطابق با **Password** کدبندی شده روی برچسب RFID باشد.

فرمان **Kill-Segmented-Memory-Tag** متغیرهای زیر را دارد:

Kill-Password (به زیربند ۷-۴-۲۶ مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

Kill-Segmented-Memory-Tag command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 25

Singulation-Id: BYTE STRING (0..255)

Kill-Password: BYTE STRING (4 bytes)

۱۰-۱۸-۲ پاسخ Kill-Segmented-Memory-Tag

پاسخ **Kill-Segmented-Memory-Tag** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Kill-Segmented-Memory-Tag response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 25

Completion-Code: INTEGER

Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
27	Zero-Kill-Password-Error
28	Kill-Failed
255	Execution-Error

Execution-Code: INTEGER

Possible Values: As defined in 9.3.

۱۰-۱۹ Delete-Packed-Object

۱۰-۱۹-۱ فرمان Delete-Packed-Object

فرمان **Delete-Packed-Object** به تحقیق‌کننده آموزش می‌دهد که **Packed-Object**ایی را حذف کند که شامل **Object-Identifier** مشخص شده، می‌باشد.

Object-Identifier به سادگی به‌عنوان نام مستعار[1] عمل می‌کند تا یک **Packed-Object** ویژه را مشخص کند.

فقط یک برچسب RFID و فقط یک **Object-Identifier** باید به ازای هر فرمان برنامه‌نویسی شود تا تضمین کند که فرآیند حذف مؤثر است. تابع حذف به خارج کردن **Packed-Object** مربوطه از برچسب و جایگزینی آن با بایت‌های لت نیاز دارد.

اگر **Packed-Object** قفل باشد، این رویه ممکن است موفق نباشد. اگر چنین موردی باشد، پاسخ، **Completion-Code** مناسب را بر می‌گرداند.

اگر پرچم **Check-Duplicate** به TRUE تنظیم شود، تحقیق‌کننده باید بررسی کند، قبل از اینکه **Packed-Object** درخواستی حذف شود، که در این صورت تنها یک نمونه از **Object-Identifier** درخواستی بر روی برچسب RFID وجود دارد. اگر تحقیق‌کننده کشف کند که برچسب RFID بیشتر از یک نمونه مرجع **Object-Identifier** را کدبندی می‌کند، نباید تابع **Delete-Packed-Object** را اجرا کند و باید **Completion-Code** مناسب را برگرداند.

اگر پرچم **Check-Duplicate** به FALSE تنظیم شود ، تحقیق‌کننده باید اولین رخداد **Packed-Object**ایی را حذف کند که با **Object-Identifier** مشخص شده است.

یادآوری- این یک متغیر است که بطور مؤثری هیچ حفاظتی را در مقابل **Object-Identifier** دو نسخه‌ای فراهم نمی‌کند. تنها باید زمانی استفاده شود که یک انتظار زیاد بدون دو نسخه‌ای وجود دارد.

فرمان **Delete-Packed-Object** متغیرهای زیر را وارد:

Check-Duplicate (به زیربند ۷-۴-۱۰ مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

Delete-Packed-Object command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 26

Singulation-Id: BYTE STRING (0..255)

Object-Identifier: OBJECT IDENTIFIER

Check-Duplicate: BOOLEAN
If set is TRUE, the interrogator shall check that there is only one occurrence of
the Object-Identifier on the RFID tag

1 - Alias

۱۰-۱۹-۲ پاسخ Delete-Packed-Object

پاسخ **Delete-Packed-Object** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Delete-Packed-Object response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 26

Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
10	Duplicate-Object
12	Object-Not-Deleted
13	Object-Identifier-Not-Found
14	Object-Locked-Could-Not-Delete
37	Data-CRC-Not-Applied
38	Length-Not-Encoded-In-DSFID
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰-۲۰ Modify-Packed-Object-Structure

۱۰-۲۰-۱ فرمان Modify-Packed-Object-Structure

فرمان **Modify-Packed-Object-Structure** استفاده می‌شود تا ساختار یک **Packed-Object** را تغییر دهد. ساختار یک **Packed-Object** ممکن است اصلاح شود تا یک نوع فهرست مشخص را تعریف کند اگر **Packed-Object** با نوع تخصیص اشاره‌گر **Packed-Object** (همان‌طور که در ۷-۴-۴۸ تعیین شده است) ایجاد شده باشد. این فرمان تعیین می‌کند که کدام نوع فهرست باید به کار برده شود. **Object-Identifier** به سادگی به‌عنوان یک نام مستعار عمل می‌کند تا یک **Packed-Object** ویژه را مشخص کند. فقط یک برچسب RFID و فقط **Object-Identifier** باید به ازای هر فرمان برنامه‌نویسی شود تا اطمینان دهد که فرآیند اصلاح مؤثر است. اگر **Packed-Object** قفل شود، این رویه ممکن است موفق نباشد. اگر این مورد وجود نداشته باشد، پاسخ **Completion-Code** مناسب را بر می‌گرداند.

تعدادی متغیر در متغیر **Packed-Object-Constructs** تعریف می‌شوند. این متغیرها باید استفاده شوند اگر وجود داشته باشد، اما اگر آن‌ها تنها به نوع **Packed-Object** مربوط شوند، با فرمان اصلاح می‌شوند. اگر آن‌ها مناسب برای **Packed-Object** فعلی نباشند، پارامترها نادیده گرفته شوند.

اگر **Packed-Object** یک نوع فهرست ویژه تعریف شده دارد، سپس **Completion-Code** مناسب برگشت داده می‌شود.

اگر پرچم **Check-Duplicate** به TRUE تنظیم شود، تحقیق‌کننده باید قبل از اینکه **Packed-Object** درخواستی را اصلاح کند، بررسی کند که تنها یک نمونه از **Object-Identifier** بر روی برچسب RFID وجود دارد. اگر تحقیق‌کننده کشف کندکه برچسب RFID بیشتر از یک نمونه مرجع **Object-Identifier** را کدبندی می‌کند، نباید تابع **Modify-Packed-Object-Structure** را اجرا کند و باید **Completion-Code** مناسب را بر گرداند.

اگر پرچم **Check-Duplicate** به FALSE تنظیم شود ، تحقیق‌کننده باید اولین رویداد **Packed-Object** را اصلاح کند که با **Object-Identifier** مشخص شده است.

یادآوری- این یک متغیر است که بطور مؤثری هیچ حفاظتی در مقابل **Object-Identifier** دو نسخه‌ای فراهم نمی‌شود. تنها باید زمانی استفاده شود که یک انتظار زیادی از بدون دو نسخه‌ای‌ها وجود دارد.

فرمان **Modify-Packed-Object-Structure** متغیرهای زیر را دارد:

Check-Duplicate (به زیربند ۷-۴-۱۰ مراجعه کنید)

Object-Identifier (به زیربند ۷-۳-۲ مراجعه کنید)

Packed-Object-Constructs (به زیربند ۷-۳-۲ مراجعه کنید)

Singulation (به زیربند ۷-۲-۱ مراجعه کنید)

Modify-Packed-Object command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 27

Singulation-Id: BYTE STRING (0..255)

Object-Identifier: OBJECT IDENTIFIER

Check-Duplicate: BOOLEAN
If set is TRUE, the interrogator shall check that there is only one occurrence of the Object-Identifier

Packed-Object-Constructs-List: [OPTIONAL] List of <Packed-Object-Constructs>

۱۰-۲۰-۲ پاسخ Modify-Packed-Object-Structure

پاسخ **Modify-Packed-Object** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Modify-Packed-Object-Structure response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 27

Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
7	Object-Locked-Could-Not-Modify
8	Singulation-Id-Not-Found
10	Duplicate-Object
13	Object-Identifier-Not-Found
30	Directory-Already-Defined
33	Insufficient-Tag-Memory
37	Data-CRC-Not-Applied
38	Length-Not-Encoded-In-DSFID
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰-۲۱ Write-Segments-6TypeD-Tag

۱۰-۲۱-۱ فرمان Write-Segments-6TypeD-Tag

فرمان **Write-Segments-6TypeD-Tag** به Data Processor در استاندارد ISO/IEC15962، آموزش می‌دهد، داده‌ها را در قطعه UII بنویسد و یا در قطعه عنوان یا هر دو قطعه بنویسد.

این فرمان ممکن است پیاده‌سازی شود تا داده‌های اولیه را در برچسب RFID بنویسد، و یا داده‌ها را به برچسب اضافه کند. این فرمان با متغیر ترکیبی **Add-Object-List** پشتیبانی می‌شود که در قطعه مبتنی بر اقلام ، به کار برده می‌شود.

این فرمان نباید استفاده شود تا یک **Momomorphic-UII** نوشته شود. اگر **AFI** بر روی برچسب RFID اعلان کند که این برای یک **Momomorphic-UII** ثبت می‌شود، خطای مناسب باید برگردانده شود و فرآیند کدبندی متوقف شود. فرمان تصحیح برای استفاده در زیربند ۱۰-۲۳ تعریف می‌شود.

اگر داده‌ایی که بر روی برچسب RFID برای اولین بار کدبندی شود، **AFI**، **UII-DSFID** و **Item-Related-DSFID** (اگر آن قطعه کدبندی شود) باید در رشته بایت کدبندی شده ترکیب شود. اگر هر یک از این سه تا، برروی برچسب از قبل کدبندی شده باشد و یک عدم انطباق با متغیر معادل در فرمان وجود داشته باشد، این رویه باید متوقف شود.

اگر **Access-Method**، **Packed-Objects** باشد، تمام **Objects** تعیین شده به عنوان متغیر، باید در همان **Packed-Objects** جدید اضافه شده قرار گیرد البته پس از آنکه هر **Packed-Object** در حافظه قرار گرفت.

چند متغیر تنها در **Packed-Object** به کاربرده می‌شود و اینها در متغیر **Packed-Object-Constructs** تعریف می‌شوند.

اگر **Access-Method**، **Tag-Data-Profiler** باشد، تمام **Objects** تعیین شده به عنوان متغیر، باید در همان **Tag-Data-Profiler** قرار گیرد. افزودن یک Object که با متغیر **Tag-Data-Profile-ID-Table** تعیین نمی‌شود باید به عنوان یک خطا رفتار شود و هیچ کدبندی روی نمی‌دهد.

چندین رکورد ممکن است تنها در قطعه مربوط به قلم کدبندی شوند. رکوردهای فردی تنها پس از ایجاد MR-header نوشته می‌شوند. جزئیات تعریف شده در زیربند ۱۰-۱۰-۱ باید در ساختار فرمان به کار برده شوند.

فرمان **Write-Segments-6TypeD-Tag** متغیرهای زیر را دارد:

AFI (به زیربند ۷-۲-۲ مراجعه کنید)

Item-Related-Add-Objects-list (به زیربند ۱۱-۶ مراجعه کنید)

Item-Related-DSFID-Constructs (به زیربند ۱۱-۷ مراجعه کنید)

Memory-Segment (به زیربند ۷-۴-۳۶ مراجعه کنید)

Multiple-Object-Constructs (به زیربند ۱۱-۱۲ مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

Tag-Data-Profile-ID-Table (به زیربند ۷-۴-۶۰ مراجعه کنید)

UII-Add-Objects-List (به زیربند ۱۱-۱۶ مراجعه کنید)

UII-DSFID-Constructs (به زیربند ۱۱-۱۷ مراجعه کنید)

Write-Segments-6TypeD-Tag command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 28

Singulation-Id: BYTE STRING (0.255)

Memory-Segment: INTEGER

Possible Values:

Value	Definition
1	UII segment
2	Item-related data segment
3	Both segments presented as objects

AFI: BYTE

Not required for **Memory-Segment** =2 (Item-related data segment)

UII-DSFID-Constructs-list: [OPTIONAL] List of <DSFID-Constructs>

Not required for Memory-Segment = 2 (Item-related data segment)

UII-Add-Objects-List: List of <Add-Objects>

Only required for Memory-Segment = 1 (UII segment), and this shall only contain one **Object-Identifier** and **Object**

Lock-UII-Segment-Arguments: INTEGER

This argument takes precedence over lock argument elsewhere in the command.

Possible Values:

Value	Definition
1	Protocol Control word
2	DSFID
3	PC word + DSFID
4	UII Data-Set, but not PC word nor DSFID
5	PC word + UII Data-Set (this combination can be applied to a Monomorphic-UII)
6	DSFID + UII Data-Set
7	The complete UII segment, including the CRC-16

Item-Related-DSFID-Constructs-list: [OPTIONAL] List of <DSFID-Constructs> Not required for Memory-Segment = 1 (UII segment)

Item-Related-Add-Objects-List: List of <Add-Objects>

Not required for Memory-Segment = 1 (UII segment)

Packed-Object-Constructs: [OPTIONAL] List of <Packed-Object-Constructs> Not required for Memory-Segment = 1 (UII segment)

Tag-Data-Profile-ID-Table: INTEGER [OPTIONAL]

Not required for Memory-Segment = 1 (UII segment)

Multiple-Records-Constructs: [OPTIONAL] List of <Multiple-Records-Constructs>

۱۰-۲۱-۲ پاسخ Write-Segments-6TypeD-Tag

پاسخ **Write-Segments-6TypeD-Tag** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Write-Segments-6TypeD-Tag response

Module-OID: OBJECT IDENTIFIER = 1 – 15961 127 28

Completion-Code: INTEGER

Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
26	AFI-Mismatch
27	DSFID-Mismatch
31	Packed-Object-ID-Table-Not-Recognised-No-Encoding
32	Tag-Data-Profile-ID-Table-Not-Recognised
33	Insufficient-Tag-Memory
36	Command-Cannot-Process-Monomorphic-UII
43	Data-Format-Not-Compatible-Multiple-Records-Header
44	Access-Method-Not-Compatible-Multiple-Records-Header
45	Sector-Identifier-Not-Compatible-Multiple-Records-Header
46	Record-Preamble-Not-Configured
47	Record-Preamble-Not-Locked
255	Execution-Error

Additional Completion-Codes apply to the Write-Response-List

Execution-Code: INTEGER

Possible Values: As defined in 9.3.

۱۰-۲۲ Read-Segments-6TypeD-Tag

۱۰-۲۲-۱ فرمان Read-Segments-6TypeD-Tag

این فرمان نباید برای خواندن هر شی داده یا بخش دیگری از یک رکورد چندگانه استفاده شود. رویه‌های مناسب در زیربند ۱۰-۲۶ تعریف می‌شوند.

فرمان **Read-Segments-6TypeD-Tag** به تحقیق‌کننده آموزش می‌دهد که یک ترتیب پیوسته‌ای از کلمات را از تمام قطعه‌های حافظه یک برچسب RFID از Type D از استاندارد ISO/IEC 18000-6 بخواند و سپس به قطعه‌ها و بخش‌های مولفه دیگر تقسیم‌بندی فرعی کند. این فرمان ممکن است برای اهداف تشخیصی مفید باشد.

هنگامی‌که پروتکل واسط هوایی می‌تواند تمام بار مفید برچسب Type D را انتقال دهد، برنامه کاربردی می‌تواند **Object-Identifier(s)** و **Object(s)** را از هر نقطه UII و یا قطعه داده مربوط به موضوع فراخوانی کند. یک

فرض اصلی این فرمان این است که تنها یک مجموعه داده **Object-Identifier** و **Object** در حافظه UII کدبندی می‌شود.

این فرمان ممکن است برای خواندن یک **Monomorphic-UII** استفاده شود که **AFI** مناسب را به عنوان یک متغیر اعلان می‌کند. Data Processor بررسی می‌کند که **AFI** به عنوان بخشی از ورودی **Monomorphic-UII** روی ثبات Data Constructs از استاندارد ISO/IEC 15961-2 می‌باشد.

- اگر این‌طور باشد، بایت‌های کدبندی شده به قواعدی فشرده می‌شوند که بر روی ثبات تعریف شده‌اند و در پاسخ گنجانده شده‌اند.
- اگر این‌طور نباشد، خطای مناسب بر گردانده می‌شود.

یادآوری- هنگامی‌که فراخوانی این فرمان که یک **Monomorphic-UII** را می‌خواند، **UII-DSFID** لازم نمی‌باشد.

متغیر **Read-Type** تنها زمانی به کار می‌رود که **Objects** از قطعه مبتنی بر اقلام بر گردانده می‌شود. این متغیر، برنامه کاربردی را قادر می‌سازد تا **Object-Identifier(s)** انتخاب شده و اعلان شده را درخواست کند تا پردازش شود و یا درخواست کند که تمام **Object-Identifier(s)** پردازش شوند.

می‌توان Memory-Segment = 4 قرار داد تا تمام بایت‌ها را بر روی قطعه UII و قطعه مبتنی بر اقلام ، به‌عنوان کدبندی شده‌اند، فراخوانی کرد، در نتیجه‌ی رشته بایت تعریف شده در استاندارد ISO/IEC 18000-6 به عنوان خروجی تحقیق‌کننده. از آنجا که همه فرآیندهای رمزگشایی توسط Data Processor استاندارد ISO/IEC 1596 قابل عبور است، رشته بایت در پاسخ می‌تواند برای هدف تشخیصی بر روی کدبندی اصلی استفاده شود.

فرمان **Read-Segments-6TypeD-Tag** متغیرهای زیر را دارد:

AFI (به زیربند ۷-۲-۲ مراجعه کنید)

Item-Related-DSFID-Constructs (به زیربند ۱۱-۷ مراجعه کنید)

Memory-Segment (به زیربند ۷-۴-۳۶ مراجعه کنید)

Read-Objects (به زیربند ۷-۴-۳۶ مراجعه کنید)

Read-Type (به زیربند ۷-۴-۵۳ مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

UII-DSFISD-Constructs (به زیربند ۱۱-۱۷ مراجعه کنید)

Read-Segments-6TypeD-Tag command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 29

Singulation-Id: BYTE STRING (0.255)

Memory-Segment: INTEGER
Possible Values:

Value	Definition
1	UII segment
2	Item-related data segment
3	Both segments presented as objects
4	Both segments as un-interpreted bytes

AFI: BYTE

UII-DSFID-Constructs-list: [OPTIONAL] List of <DSFID-Constructs> Not required for Memory-Segment = 2 (Item-related data segment) Not required if the AFI is for a monomorphic-UII

Item-Related-DSFID-Constructs-list: [OPTIONAL] List of <DSFID-Constructs> Not required for Memory-Segment = 1 (UII segment)

Read-Type: INTEGER
Not required for Memory-Segment = 1 (UII segment)
Possible Values:

Value	Definition
1	Read-Multiple-Objects
2	Read-All-Objects

Read-Objects-List: List of <Read-Objects> This does not apply to Read-All-Objects (2)

۱۰-۲۲-۲ پاسخ Read-Segments-6TypeD-Tag

پاسخ **Read-Segments-6TypeD-Tag** متغیرهای زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

ISO-UII memory (به زیربند ۱۱-۵ مراجعه کنید)

Length-Lock Byte (به زیربند ۷-۵-۳ مراجعه کنید)

Read-Data (به زیربند ۷-۵-۱۲ مراجعه کنید)

Read-Objects-Response-List (به زیربند ۱۱-۱۴ مراجعه کنید)

Read-Segments-6TypeD-Tag response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 29

ISO-UII Memory-List: List of <ISO-UIIMemory>
This response argument is included when the content of the UII segment has been requested

Length-Lock Byte: BYTE
This response argument is included when the content of the item-related data
segment has been requested

Read-Objects-Response-List: List of <Read-Objects-Response>
This response argument is included when the content of the item-related data
segment has been requested

Read-Data: BYTE STRING
This response argument is included when the Memory-Segment = 4 (Both segments as un-interpreted bytes)

Completion Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰-۲۳ Write-Monomorphic-UII

۱۰-۲۳-۱ فرمان UI Write-Monomorphic-UII

فرمان **Write-Monomorphic-UII** به Data Processor در استاندارد ISO/IEC 15962 آموزش می‌دهد تا یک **Monomorphic-UII** را در ابتدا بنویسد و یا یک **Monomorphic-UII** موجود را اصلاح کند. متغیرها بطور انتخابی با توجه به نوع برچسب RFID که مشخص می‌شود به کاربرده می‌شوند.

یک **Monomorphic-UII** تنها با یک **AFI** کدبندی شده پشتیبانی می‌شود و هیچ **DSFID**یی کدبندی نمی‌شود. **Object-Identifier** تنها برای اهداف ارتباط بین برنامه کاربردی و Data Processor لازم است. بر روی برچسب RFID کدبندی نمی‌شود. تنها **Compact-Parameter** با مقدار ۵ (**Monomorphic-UII**) باید در فرمان استفاده شود.

فرآیند کدبندی کلی، Data Processor را فراخوانی می‌کند تا بررسی کند که **AFI** در فرمان، مطابق با یک **Monomorphic-UII** روی ثبات Data Constructs از استاندارد ISO/IEC 15961-2 می‌باشد. اگر یک انطباق امکان‌پذیر نباشد یا به دلیل اینکه **AFI** برای یک **Monomorphic-UII** ثبت نمی‌شود و یا اینکه هیچ **AFI** انطباقی را نمی‌توان بر روی ثبات پیدا کرد آنگاه فرآیند متوقف می‌شود.

اگر **AFI** مطابقت داشته باشد، **Object-Identifier** نیز برای یک انطباق با ثبات Data Constructs از استاندارد ISO/IEC 15961-2، بررسی می‌شود. یک عدم انطباق یک **Completion-Code** مناسبی را تولید می‌کند، اما این تنها باید به عنوان یک هشدار در مورد ساخت فرمان بحث شود. این فرآیند ادامه دارد زیرا **Object-Identifier** کدبندی نمی‌شود. Data Processor از طرح فشرده‌سازی مربوط به **AFI** روی ثبات Data Constructs از استاندارد ISO/IEC 15961-2، استفاده می‌کند که به‌طورصریح تعریف شده است تا **Monomorphic-UII** را کدبندی کنند.

اگر **Memory-Type**، (18000-6 Type C MB01) 0 باشد، اولین شرط لازم خواندن محتوای MB01 می‌باشد تا همه کدبندی‌های موجود را برقرار کنند. اگر این قفل شود، فرآیند متوقف می‌شود، در غیر این‌صورت این فرآیند ادامه دارد. فرآیند کدبندی **Object** را فشرده می‌کند و اگر این منجر تعداد فردی از بایت‌ها شود، بایت خاتمه‌دهنده[1] 00_{16} اضافه‌می‌شود[2]. کلمه «کنترل پروتکل»[3] ساخته می‌شود که **AFI** و بیت‌های طول را ترکیب کند. اگر **Access-Password** در فرمان باشد برای انطباق آن بر روی برچسب RFID استفاده می‌شود تا در مقابل نوشتن غیرمجاز داده‌ها در برچسب RFID حفاظت کند. اگر این فرمان برای رونویسی[4] بر روی MB01 با یک **Monomorphic-UII** جدید استفاده شود، لازم است که طول رشته بایت جاری و رشته بایت جدید مقایسه کند. اگر کد جدید طول کوتاه‌تری داشته باشد، تحقیق‌کننده باید مطمئن شود که بایت‌هایی که قسمتی از کد قدیمی را نشان می‌دهد با بایت‌های صفر رونویسی شده است.

اگر **Memory-Type**، (قطعه 18000-6 type D UII) ۱ باشد، اولین الزام، خواندن محتوای قطعه UII می‌باشد تا هر کدبندی موجود را ایجاد کند. اگر این قفل باشد، فرآیند متوقف می‌شود، در غیراین‌صورت این فرآیند ادامه دارد. فرآیند کدبندی **Object** را فشرده می‌کند و اگر این منجر به تعداد فردی از بایت‌ها شود، بایت پایانی 00_{16} را اضافه می‌کند. کلمه Protocol Control ساخته می‌شود، **AFI** و بیت‌های طول را با هم ترکیب می‌کند. اگر این فرمان برای رونویسی قطعه UII با یک **Monomorphic-UII** جدید استفاده شود، لازم است که طول رشته بایت جاری و رشته بایت جدید مقایسه کند. اگر کد جدید طول متفاوتی در **Monomorphic-UII** موجود (یعنی کوتاه‌تر یا بلندتر) داشته باشد، تحقیق‌کننده باید بررسی کند آیا یک قطعه مربوط به عنوان کدبندی می‌شود. دستورالعمل رونویسی، همان‌طور که در ISO/IEC 15962 تعریف شده است باید در نظر بگیرد که آیا

1 - Terminator
2 - Append
3 - Protocol Control
4 - Over-Write

داده‌های مربوط به عنوان روی RFID وجود دارد و آیا هر یک از اینها قفل می‌باشد. این لازم است زیرا کدبندی بر روی یک برچسب Type D از استاندارد ISO/IEC 18000-6 بین قطعه‌ها پیوسته است.

اگر **Memory-Type**، ۲ (برچسب حافظه تک) باشد، اولین الزام، خواندن **AFI** بر روی برچسب RFID (که در یک ناحیه حافظه جدا می‌تواند کدبندی شود) و حداقل ۱۶ بایت از محتوای حافظه کاربر می‌باشد تا هر کدبندی موجود برقرار شود. اگر بایت‌های کدبندی شده پیدا شوند، سپس این فرآیند ادامه می‌یابد تا اینکه یک رشته‌ای از چهار بایت صفر پیدا شود. اگر **AFI** یا هر بخش از حافظه کاربر قفل شود این فرآیند متوقف می‌شود، در غیر این‌صورت این فرآیند ادامه دارد. فرآیند کدبندی، **Object** را فشرده می‌کند و طول **Monomorphic-UII** فشرده را به عنوان یک پیشوند اضافه می‌کند. اگر این فرمان، برای رونویسی با یک **Monomorphic-UII** استفاده شود لازم است که طول جریان و رشته بایت جدید را مقایسه کند. اگر کد جدید طول کوتاه‌تری داشته باشد، تحقیق‌کننده باید مطمئن شود که بایت‌هایی که بخشی از کد قدیمی‌تر را نشان می‌دهد با بایت‌های صفر رونویسی می‌شوند. اگر **AFI** صحیح برای **Monomorphic-UII**، تاکنون بر روی برچسب RFID کدبندی نشده است، هم‌اکنون کدبندی می‌شود.

فرمان **write-Monomorphic-UII** متغیرهای زیر را دارد:

Access-Password (به زیربند ۷-۴-۱ مراجعه کنید)

AFI (به زیربند ۷-۲-۲ مراجعه کنید)

AFI-Lock (به زیربند ۷-۴-۳ مراجعه کنید)

Compact-Parameter (به زیربند ۷-۳-۵ مراجعه کنید)

Memory-Bank-Lock (به زیربند ۷-۴-۳۴ مراجعه کنید)

Memory-Type (به زیربند ۷-۴-۳۷ مراجعه کنید)

Object (به زیربند ۷-۳-۴ مراجعه کنید)

Object-Identifier (به زیربند ۷-۳-۲ مراجعه کنید)

Object-Lock (به زیربند ۷-۳-۶ مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

Write-Monomorphic-UII command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 30

Singulation-Id: BYTE STRING (0..255)

Memory-Type: INTEGER (0..15)

Possible Values:

Value	Definition
0	18000-6 Type C MB01
1	18000-6 Type D UII segment
2	single memory tag

Access-Password: [OPTIONAL] BYTE STRING (4 bytes)

This may only be applied for **Memory-Type** 0 (18000-6 Type C MB01). Additionally it is optional.

AFI: BYTE

AFI-Lock: BOOLEAN

This may only be applied for **Memory-Type** 2 (single memory tag) If set to TRUE, the interrogator shall lock the AFI

Object-Identifier: OBJECT IDENTIFIER

This is the **Object-Identifier** associated with the AFI on the ISO/IEC 15961 Data Constructs register.

Object: BYTE STRING

This is the Monomorphic-UII as presented by the application

Compact-Parameter: INTEGER

The only permitted value is 5 (Monomorphic-UII), which shall cause the Data Processor to call the explicitly defined compaction scheme associated with the AFI on the ISO/IEC 15961 Data Constructs register.

Object-Lock: BOOLEAN

This may only be applied for **Memory-Type** 2 (single memory tag)
If TRUE the interrogator shall lock the relevant Data-Set

Memory-Bank-Lock: BOOLEAN

This may only be applied for **Memory-Type** 0 (18000-6 Type C MB01) If TRUE the entire memory bank shall be locked.

Lock-UII-Segment-Arguments: INTEGER

This may only be applied for **Memory-Type** 1 (18000-6 Type D UII segment)

Possible Values:

Value	Definition
4	UII Data-Set, but not PC word nor DSFID
5	PC word + UII Data-Set (this combination can be applied to a Monomorphic-UII)
7	The complete UII segment, including the CRC-16

Not required for Memory-Segment = 1 (UII segment)

Multiple-Records-Constructs: [OPTIONAL] List of <Multiple-Records-Constructs>

۱۰-۲۳-۲ پاسخ Write-Monomorphic-UII

پاسخ **Write-Monomorphic-UII** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Write-Monomorphic-UII response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 30

Completion-Code: INTEGER

Possible Values:

Value	Definition
0	No-Error
7	Object-Locked-Could-Not-Modify
8	Singulation-Id-Not-Found
22	Object-Modified-But-Not-Locked
25	Password-Mismatch
26	AFI-Mismatch
33	Insufficient-Tag-Memory
34	AFI-Not-For-Monomorphic-UII
35	Monomorphic-UII-OID-Mismatch
255	Execution-Error

Execution-Code: INTEGER

Possible Values: As defined in 9.3.

۱۰-۲۴ Configure-Extended-DSFID

فرمان **Configure-DSFID** (به زیربند ۱۰-۲ مراجعه کنید) ساختاری است که از کدبندی **Access-Method** و یک **Data-Format** پشتیبانی می‌کند. از زمان انتشار استاندارد ISO/IEC 15961:2004، ویژگی‌های دیگری اضافه شدند تا یک **DSFID** توسعه یافته را بسازند، برای مثال، علامت‌دهی طول داده‌های کدبندی شده یا اینکه آیا کشف خطای داده‌ها به داده‌هایی اضافه شده است که برای یک مدت طولانی ذخیره می‌شوند. فرمان **Configure-Extended-DSFID** استفاده می‌شود تا نشانگرهای مختلف را که درون **Extended-DSFID** کدبندی شده‌اند، تنظیم کند. زمانی که **DSFID** تعیین می‌کند کدام قاعده کدبندی را دنبال کند، ویژگی‌های اضافی اعلان شده با این فرمان، به آسانی فرآیندهای اضافه‌ای را تعیین می‌کنند که زمان کدبندی داده‌ها استفاده می‌شوند.

۱۰-۲۴-۱ فرمان Configure-Extended-DSFID

فرمان **Configure-Extended-DSFID** برای آموزش Data Processor استفاده می‌شود تا تمام ویژگی‌های **DSFID** مناسب را بر روی برچسب RFID کدبندی کنند. این فرمان باید قبل از اینکه هر کدبندی روی برچسب RFID روی دهد، فراخوانی شود.

DSFID-Configure-List استفاده می‌شود تا **Access-Method** و **Data-Format** را تعیین کند. **Ext-DSFID-Constructs-list** برای قرار دادن نشانگرها روی **Extended-DSFID** و برای آموزش Data Processor استفاده می‌شود تا رویه‌های معینی را انجام دهند، مثلا، استفاده از یک CRC در داده‌ها.

متغیر **DSFID-Lock**، اگر تنظیم شود، در تمام رشته بایت **DSFID** و **Extended-DSFID** به کاربرده می‌شود.

فرمان **Configure- DSFID** متغیرهای زیر را دارد:

DSFID-Constructs (به زیربند ۱۱-۲ مراجعه کنید)

DSFID-Lock (به 0 مراجعه کنید)

Ext-DSFID-Constructs (به زیربند ۱۱-۴ مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

متغیر **DSFID-Lock** باید تنها زمانی به کاربرده شود که کاربر تصمیم می‌گیرد تمام ویژگی‌های **Extended-DSFID** بتوانند بطوردائم کدبندی شوند.

Configure-Extended-DSFID command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 31

Singulation-Id: BYTE STRING (0..255)

DSFID-Constructs-list: List of <DSFID-Constructs>

Ext-DSFID-Constructs-list: [Optional] List of <Ext-DSFID-Constructs>

DSFID-Lock: BOOLEAN
If set to TRUE, the interrogator shall lock the DSFID

۱۰-۲۴-۲ پاسخ Configure-Extended-DSFID

پاسخ **Configure-Extended-DSFID** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Configure-Extended-DSFID response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 31

Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
4	DSFID-Not-Configured
5	DSFID-Not-Configured-Locked
6	DSFID-Configured-Lock-Failed
8	Singulation-Id-Not-Found
37	Data-CRC-Not-Applied
38	Length-Not-Encoded-In-DSFID
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰-۲۵ Configure-Multiple-Records-Header

فرمان **Configure-Multiple-Records-Header** باید اولین فرمانی باشد که درخواست شده است تا یک برچسب RFID فعال شود تا از چند رکورد پشتیبانی کند. این فرمان را می‌توان در هر برچسب با ظرفیت حافظه کافی به کار برد، اما تنها در Memory Bank 11 از یک برچسب حافظه قطعه‌بندی شده به کار برده می‌شود، از قبیل Type C از استاندارد ISO/IEC 18000-6.

۱۰-۲۵-۱ فرمان Configure-Multiple-Records-Header

فرمان **Configure-Multiple-Records-Header** برای آموزش Data Processor استفاده می‌شود تا تمام ویژگی‌های مناسب MR-header را برروی برچسب RFID کدبندی کند. این فرمان باید درخواست شود قبل از اینکه هر کدبندی از یک رکورد، روی برچسب RFID قرار گیرد.

Access-Password در این فرمان، استفاده می‌شود تا آن را روی برچسب RFID تطبیق دهد تا به نوشتن داده‌ها در برچسب RFID اجازه دهد. این تنها در برچسب Type C از استاندارد ISO/IEC 18000-6 به کار برده می‌شود و همچنین اگر چنین کلمه عبوری از قبل کدبندی شده باشد، می‌توان در آن برچسب RFID به کار برد.

DSFID-Configure-List استفاده می‌شود تا (=4) **Access-Method** و **Data-Format** را مشخص کند. اگر تمام رکوردها معلوم باشند که یک **Data-Format** همانند و مشترکی دارند (که به‌عنوان کدبندی **Multiple-Records** همگن تعریف می‌شود)، سپس مقدار آن **Data-Format** باید کدبندی شود. اگر هر یک از رکوردها **Data-Format** مختلفی داشته باشند (که به‌عنوان کدبندی **Multiple-Records** ناهمگن تعریف می‌شود)، سپس {00001} **Data-Format** باید استفاده شود.

یادآوری- برای **Multiple-Records Access-Method**، تفسیر {00001} **Data-Format** این است که رکوردها یک **Data-Format** مشترکی ندارند.

زیرا {4} **Access-Method** لازم است، متغیر **Ext-DSFID-Constructs-list** (به زیربند ۱۱-۴ مراجعه کنید) لازم است که استفاده شود، اما با این متغیرهای ویژه:

- **Memory-Length-Encoding** اختیاری است و در فهرست به کار برده می‌شود، مقادیر ممکن برای رشته بیت ۰۰ هستند تا نشان دهند که هیچ کدبندی درخواست نمی‌شود و ۱۰ نشان دهنده این است که طول فعلی فهرست باید با Data Processor محاسبه و کدبندی شود. و در MR-header کدبندی شود. اگر MR-header قفل شود، سپس این را نمی‌توان استفاده کرد.
- **Data-CRC-Indicator** اختیاری است و در تمام مولفه‌های فهرست استفاده می‌شود، اما زمانی توصیه می‌شود که یک فهرست وجود دارد. تنها مقدار ممکن برای رشته بیت ۱۰ است.
- **Simple-Sensor-Indicator** شرطی است و تنها زمانی لازم است که سازوکارهای دیگر روی برچسب RFID وجود ندارد تا وجود حسگرهای ساده را اعلان کنند.
- **Battery-Assist-Indicator** شرطی است و تنها زمانی لازم است که اگر سازوکارهای دیگر روی برچسب RFID وجود نداشته باشند وجود یک باطری را اعلان کنند.
- **Full-Function-Sensor-Indicator** شرطی است و تنها زمانی لازم است که سازوکارهای دیگر بر روی برچسب RFID وجود ندارند تا وجود حسگرهای تابع کامل را اعلان کنند.
- **DSFID-Pad-Bytes** نباید استفاده شود به دلیل اینکه به راحتی بایت‌های لت را برای MR-header کامل، اضافه می‌کنند.

Directory-Length-EBV8-Indecator (به زیربند ۷-۴-۱۴ مراجعه کنید) اجباری است و به Data Processor آموزش می‌دهد تا فضای کافی را در MR-header فراهم کند تا اندازه فهرست در حدود اندازه‌ای کدبندی شود که با برنامه کاربردی تعیین شده است.

Multiple-Records-Feature-Indicator اجباری است و یک نگاشت بیت می‌باشد تا اطلاعات بیشتری را در مورد رکوردهای کدبندی شده در Logical Memory فراهم کند. (برای جزئیات بیشتر به زیربند ۷-۴-۳۹ مراجعه کنید)

Sector-Identifier (به زیربند ۷-۴-۵۷ مراجعه کنید) زمانی که در این متغیر به‌عنوان یک مقدار غیر صفر قرار می‌گیرد شناسه بخش صحیح را نشان می‌دهد، اما مقدار صفر نشان می‌دهد که شناسه بخش صحیح در هر یک از رکوردهای فردی کدبندی می‌شود.

Pointer-To-Multiple-Records-Directory (به زیربند ۷-۴-۵۱ مراجعه کنید) یک آدرس EBV-8 می‌باشد که شروع فهرست را نشان می‌دهد. مقادیر ویژه از پاسخ‌های واسط هوایی حاصل می‌شوند که جزئیات نگاشت حافظ سخت‌افزاری برچسب RFID را فراهم می‌کنند.

متغیر **Lock-Multiple-Records-Header** (به زیربند ۷-۴-۲۹ مراجعه کنید) استفاده می‌شود تا تعیین کند آیا کل MR-header قفل است یا نه. اگر قفل نباشد، گزینه‌های زیر امکان‌پذیر هستند:

- تنها طول داده از فیلد فهرست را قفل‌گشایی شده می‌گذارند.
- تنها تعدادی از فیلدهای رکورد را قفل‌گشایی شده می‌گذارند.
- هم طول داده از فیلد فهرست و تعداد فیلد رکوردها را قفل‌گشایی شده می‌گذارند.

در موارد قفل‌گشایی شده، Data Processor باید اطمینان دهد که بایت‌های لت لازم پس از رشته‌های قبلی اضافه می‌شوند تا تنظیم بلوک قبل از قفل شدن آن‌ها به دست آید. همچنین باید رشته‌های قفل‌گشایی شده را بلوک ردیف کنند.

فرمان **Configure-Multiple-Records-Header** متغیرهای زیر را دارد:

Access-Password (به زیربند ۷-۴-۱ مراجعه کنید)

Directory-Length-EBV8-Indicator (به زیربند ۷-۴-۱۴ مراجعه کنید)

DSFID-Constructs (به زیربند ۱۱-۲ مراجعه کنید)

Ext-DSFID-Constructs (به زیربند ۱۱-۴ مراجعه کنید)

Lock-Multiple-Records-Header (به زیربند ۷-۴-۳۹ مراجعه کنید)

Multiple-Records-Features-Indicator (به زیربند ۷-۴-۳۹ مراجعه کنید)

Pointer-To-Multiple-Records-Directory (به زیربند ۷-۴-۵۱ مراجعه کنید)

Sector-Identifier (به زیربند ۷-۴-۵۷ مراجعه کنید)

Singulation-Id (به۷-۲-۱ مراجعه کنید)

Configure-Multiple-Records-Header command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 32

Singulation-Id: BYTE STRING (0..255)

Access-Password: [OPTIONAL] BYTE STRING (4 bytes)

DSFID-Constructs-list: List of <DSFID-Constructs>

Ext-DSFID-Constructs-list: List of <Ext-DSFID-Constructs>

Directory-Length-EBV8-Indicator: [Optional] INTEGER
Possible Values:

Value	Definition
1	Single byte EBV-8
2	Double byte EBV-8, even if the length is less than 128 blocks

Multiple-Records-Features-Indicator: BYTE
This is a bit map that is set in this command that determines rules for the Data Processor to follow when encoding individual records.

Sector-Identifier:INTEGER
Possible Values:

Value	Definition
0	Indicates that the sector identifier varies between records, and that the true value is only obtainable for the individual record.
≠0	Indicates the true value of the sector that applies to all records.

Pointer-To-Multiple-Records-Directory: EBV-8 VALUE
This value is based on responses from the interrogator on the memory mapping of the RFID tag that is being addressed.

Lock-Multiple-Records-Header: INTEGER
Possible Values:

Value	Definition
0	Not locked
1	Completely locked
2	All components locked except for the Number of records field.

۱۰-۲۵-۲ پاسخ Configure-Multiple-Records-Header

پاسخ **Configure-Multiple-Records-Header** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Configure-Multiple-Records-Header response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 32

Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
4	DSFID-Not-Configured
8	Singulation-Id-Not-Found
25	Password-Mismatch
37	Data-CRC-Not-Applied
38	Length-Not-Encoded-In-DSFID
39	Multiple-Records-Header-Not-Configured
40	Multiple-Records-Header-Not-Locked
41	File-Support-Indicators-Not-Configured
42	File-Support-Indicators-Not-Locked
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰- ۲۶ Read-Multiple-Records

۱۰-۲۶-۱ فرمان Read-Multiple-Records

فرمان **Read-Multiple-Records** به تحقیق‌کننده آموزش می‌دهد که مولفه‌های مختلفی را با ساختار منطقی از یک برچسب RFID بخواند که پیکربندی شده است تا چند رکورد را کدبندی کنند. این شامل خواندن مجموعه‌ای از یک یا چند **Object-Identifiers** و **Objects** مربوطه از یک رکورد فردی می‌باشد. تنها یک برچسب RFID باید در هر فرمان برنامه‌نویسی شود تا مطمئن شویم که فرآیند خواندن مؤثر است.

این فرمان در بانک حافظه ۱۱ از برچسب Type C از استاندارد ISO/IEC 18000-6 و به قطعه داده مبتنی بر قلم از برچسب Type D از استاندارد ISO/IEC 18000-6 به کار برده می‌شود.

متغیر **Read-Record-Type** (به زیربند ۷-۴-۵۲ مراجعه کنید) مجموعه‌ای از گزینه‌ها را برای خواندن داده‌ها از برچسب تهیه می‌کند.

فرمان **Read-Multiple-Records** متغیرهای زیر را دارد:

Max-App-Length (به زیربند ۷-۴-۳۲ مراجعه کنید)

Read-Objects (به زیربند ۱۱-۱۳ مراجعه کنید)

Read-Record-Type (به زیربند ۷-۴-۵۲ مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

Read-Multiple-Records command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 33

Singulation-Id: BYTE STRING (0..255)

Read-Record-Type: INTEGER

Possible Values:

Value	Definition
0	Read-Multiple-Records-Header
1	Read-Multiple-Records-Header-Plus-1st-Preamble
2	Read-Multiple-Records-Directory
3	Read-Preamble-Specific-Multiple-Record
4	Read-All-Record-OIDs-Specific-Record-Type
5	Read-OIDs-Specific-Multiple-Record
6	Read-All-Objects-Specific-Multiple-Record
7	Read-Multiple-Objects-Specific-Multiple-Record
8	Read-1st-Objects-Specific-Multiple-Record
9	Read-Data-Element-List-Specific-Multiple-Record

Read-Objects-List: List of <Read-Objects>

This only applies to Read-Preamble-Specific-Multiple-Record (3), Read-All-Record-OIDs-Specific- Record-Type (4), Read-All-Objects-Specific-Multiple-Record (6), Read-Multiple-Objects-Specific- Multiple-Record (7), Read-1st-Objects-Specific-Multiple-Record (8), and Read-Data-Element-List-Specific-Multiple-Record (9)

See 7.4.52 for details of what **Objects** are relevant to each **Read-Record-Type**

Max-App-Length: INTEGER (1..65535)

This only applies to Read-Type (8) and is expressed in bytes

۱۰-۲۶-۲ پاسخ Read-Multiple-Records

پاسخ **Read-Multiple-Records** متغیر زیر و نام‌های فیلد را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Multiple-Records-Director-structure-list (به زیربند ۱۱-۹ مراجعه کنید)

Multiple-Records-Preamble-Structure (به زیربند ۱۱-۱۱ مراجعه کنید)

Read-Objects-Response-list (به زیربند ۱۱-۱۴ مراجعه کنید)

Read-OIDs-Response-list (به 0 مراجعه کنید)

Read-Multiple-Records response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 33

Multiple-Records-Header-Structure-List: [Conditional] List of <Multiple-Records-Header-Structure>

Multiple-Records-Preamble-Structure: [Conditional] List of <Multiple-records>

Multiple-Records-Directory-Structure-List: [Conditional] List of <Multiple-Records-Directory-Structure>

Read-OIDs-Response-List: [Conditional] List of <Read-OIDs-Response>

Read-Objects-Response-List: [Conditional] List of <Read-Objects-Response>
Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
48	Multiple-Records-Directory-Not-Present
255	Execution-Error

Additional Completion-Codes apply to the: Read-Objects-Response-List

Execution-Code: INTEGER
Possible Values: As defined in 9.3.

۱۰-۲۷ Delete-Multiple-Record

۱۰-۲۷-۱ فرمان Delete-Multiple-Record

فرمان **Delete-Multiple-Record** به تحقیق‌کننده آموزش می‌دهد که یک **Multiple-Record** را به‌عنوان حذف شده علامت‌گذاری کند و یا بطور فیزیکی رکورد را حذف کند.

متغیر **Delete-MR-Method** در این فرمان، هنگامی‌که از این دو روش استفاده می‌شود، به تحقیق‌کننده آموزش می‌دهد.

اگر روش، رکورد را به‌عنوان حذف شده علامت‌گذاری کند، بایت‌هایی که رکورد را تشکیل می‌دهند در واقع حذف نمی‌شوند، اما رکورد موضوع و ورودی آن در فهرست (اگر وجود داشته باشد)، مقادیر کدی را دارند تا نشان دهند که این رکورد دیگر معتبر نیست. اگر مقدمه رکورد و یا فهرست قفل شود، آنگاه فرمان نمی‌تواند فراخوانی شود. اگر این رکورد بخشی از یک ساختار سلسله مراتبی باشد، آنگاه تمام رکوردهای فرزند از پایین‌ترین سطح

باید ابتدا حذف شوند. اگر هر یک از این مقدمه‌های رکورد و یا ورودی فهرستی مربوطه قفل شود آنگاه هیچ‌کدام از این رکوردها را نمی‌توان حذف کرد.

اگر این روش بطور فیزیکی رکورد را حذف کند، سپس تمام کدبندی در رکورد تغییر می‌کند تا این بخش از حافظه را برای کدبندی آینده در دسترس قرار دهند. این روش را نمی‌توان استفاده کرد اگر هر بخشی از رکورد قفل شود، اما اگر مقدمه رکورد قفل نشود، آنگاه روش دیگری ممکن است امتحان شود (علامت‌گذاری رکورد به‌عنوان حذف شده).

Access-Password در این فرمان، استفاده می‌شود تا آن را بر روی برچسب RFID تطابق دهد (به منظور اجازه برای نوشتن داده‌ها در برچسب RFID) و حذف به مفهوم یک تراکنش نوشتن است. این تنها در برچسب Type C از استاندارد ISO/IEC 18000-6 به کار برده می‌شود و اگر چنین کلمه عبوری از قبل کدبندی شده باشد، تنها در برچسب RFID به کار برده می‌شود.

فرمان **Delete-Multiple-Record** متغیرهای زیر را دارد:

Access-Password (به زیربند ۷-۴-۱ مراجعه کنید)

Object-Identifier (به زیربند ۷-۳-۲ مراجعه کنید)

Singulation-Id (به زیربند ۷-۲-۱ مراجعه کنید)

Delete-Multiple-Record command

Module-OID: OBJECT IDENTIFIER = 1 0 15961 126 34

Singulation-Id: BYTE STRING (0..255)

Access-Password: [OPTIONAL] BYTE STRING (4 bytes)

Delete-MR-Method: INTEGER

Possible Values:

Value	Definition
0	Mark record as deleted
1	Physically delete to record

Object-Identifier: OBJECT IDENTIFIER

This is the **Object-Identifier** to the level where the final arc is that of the instance-of or hierarchical identifier

۱۰-۲۷-۲ پاسخ Delete-Multiple-Record

پاسخ **Read-Multiple-Records** نام‌های فیلد زیر را دارد:

Completion-Code (به زیربند ۹-۲ مراجعه کنید)

Execution-Code (به زیربند ۹-۳ مراجعه کنید)

Delete-Multiple-Record response

Module-OID: OBJECT IDENTIFIER = 1 0 15961 127 34

Completion-Code: INTEGER
Possible Values:

Value	Definition
0	No-Error
8	Singulation-Id-Not-Found
13	Object-Identifier-Not-Found
25	Password-Mismatch
49	Record-Not-Deleted-Preamble-Locked
50	Record-Not-Deleted-Directory-Locked
51	Record-Not-Deleted-Lower-Level-Preamble-Locked
52	Record-Not-Deleted-Encoding-Locked
255	Execution-Error

Execution-Code: INTEGER
Possible Values: As defined in 9.3

۱۱ متغیرها

۱۱-۱ Add-Objects

این متغیر فهرستی از **Object-Identifier** و **Objects** را فراهم می‌کند که در RFID نوشته می‌شوند، یا در یک برچسب «خالی[1]» و یا افزودن به داده‌ای که از قبل کدبندی شده است.

متغیر **Add-Objects** متغیرهای زیر را دارد:

Avoid-Duplicate (به زیربند ۷-۴-۶ متغیر مراجعه کنید)

Compact-Parameter (به زیربند ۷-۳-۵ مراجعه کنید)

Object (به زیربند ۷-۳-۴ مراجعه کنید)

Object-Identifier (به زیربند ۷-۳-۲ مراجعه کنید)

Object-Lock (به زیربند ۷-۳-۶ مراجعه کنید)

1 - Blank

Add-Objects argument

Object-Identifier: OBJECT IDENTIFIER

Avoid-Duplicate: BOOLEAN
If TRUE, the interrogator shall check that the subject Object-Identifier
is not already encoded on the RFID tag.

Object: BYTE STRING

Compact-Parameter: INTEGER (0..15)
Possible Values:

Value	Definition (see 7.3.5 for further details)
0	Application-Defined
1	Compact
2	UTF8-Data
3	Packed-Objects
4	Tag-Data-Profile

Object-Lock: BOOLEAN
If TRUE the interrogator shall lock the relevant Data-Set

اگر **Compact-Parameter**، **Packed-Objects** باشد، آنگاه باید برای همه مولفه‌ها در فهرست یکسان باشد؛ و مقدار **Object-Lock** نیز باید در سراسر فهرست، همسان باشد.

DSFID-Constructs ۲-۱۱

DSFID-Constructs argument

Access-Method: INTEGER

Possible Values

Value	Definition
0	No-Directory
1	Directory
2	Packed-Objects
3	Tag-Data-Profile
4	Multiple-Records
5 to 15	Reserved for extension when extensions to DSFID are Implemented

Data-Format: INTEGER

Possible Values

Value	Definition
0	Not-Formatted
1	Full-Featured
2	Root-OID-Encoded
3 to 28	As published by the Registration Authority of ISO/IEC 15961-2
29	For closed systems compliant with ISO/ IEC 15962
30	For closed systems with proprietary encoding
31	Not applicable as an assigned DSFID
32 to 287	Reserved as an extension for multiple byte Data-Format

EPC-UII memory ۳-۱۱

EPC-UII-Memory argument

Protocol-Control-Word: BYTE STRING (2)

EPC-Code: BYTE STRING

Ext-DSFID-Constructs ۴-۱۱

این متغیر فهرستی از متغیرهای اضافی را فراهم می‌کند که استفاده می‌شوند تا ویژگی‌های اضافی را بر روی برچسب نشان دهند و یا فرآیندهایی که Data Processor باید استفاده کند تا کدبندی را بر روی برچسب RFID کامل کند. این متغیر نیز برای برخی از پاسخ‌ها استفاده می‌شود.

اگر هر کدام یا هر دو بیت **Memory-Length-Encoding** = 1 باشد، Data Processor باید تعداد مناسب بلوک‌ها را محاسبه کند و این را بر روی برچسب RFID کدبندی کند که سازگار با قواعد نحوی Extended DSFID باشد.

اگر هر یک یا هر دو بیت **Data-CRC-Indicator** = 1 باشد، Data Processor باید CRC داده‌های مناسب را محاسبه کند و این را در موقعیت‌های مناسب در حافظه کدبندی کند.

اگر برنامه کاربردی تعیین کند که **Battery-Assist-Indicator**، و/یا **Full-Function-Sensor-Indicator** و/یا **Simple-Sensor-Indicator** باید به ۱(TRUE) تنظیم شود، Data Processor باید این تنظیمات را کدبندی کند که آیا ویژگی انتخاب شده واقعا توسط برچسب RFID پشتیبانی می‌شود یا نه.

بایت‌های اضافی ممکن است در **Extended-DSFID** برای کدبندی آینده بر روی برچسب RFID، توسط تعیین تعداد **DSFID-Pad-Bytes** کدبندی شوند.

متغیرهای **Length-of-Encoded-Data** و **Memory-Capacity**، هنگامی‌که **Extended-DSFID-Constructs** در یک پاسخ قرار می‌گیرد، به عنوان متغیرها بر گردانده می‌شوند.

متغیر **Extended-DSFID-Constructs** متغیرهای زیر را دارد:

Battery-Assist-Indicator (به زیربند ۷-۴-۷ مراجعه کنید)

Data-CRC-Indicator (به زیربند ۷-۴-۱۱ مراجعه کنید)

DSFID-Pad-Bytes (به زیربند ۷-۴-۱۶ مراجعه کنید)

Full-Function-Sensor-Indicator (به زیربند ۷-۴-۲۰ مراجعه کنید)

Length-of-EnCoded-Data (به زیربند ۷-۵-۴ مراجعه کنید)

Memory-Capacity (به زیربند ۷-۵-۷ مراجعه کنید)

Memory-Length-Encoding (به زیربند ۷-۴-۳۵ مراجعه کنید)

Simple-Sensor-Indicator (به زیربند ۷-۴-۵۸ مراجعه کنید)

Ext-DSFID-Constructs argument

Memory-Length-Encoding: BIT STRING

Possible Values

Value	Definition
00	No encoded length or memory capacity is small (i.e. not more than 256 bits)
01	Memory capacity is defined
10	The length of encoded data is defined
11	Both memory capacity and length of encoding are defined

Data-CRC-Indicator: BIT STRING

Possible Values

Value	Definition
00	No data CRC
01	Data CRC applied to each individual data set
10	Data CRC applied only to the entire encoded data
11	Data CRC applied to each data set and to the entire encoded data

Simple-Sensor-Indicator: BOOLEAN
If a simple sensor is on the tag, this is set as TRUE.

Battery-Assist-Indicator: BOOLEAN
If this is a battery assisted RFID tag, this is set as TRUE.

Full-Function-Sensor-Indicator: BOOLEAN
If a full-function sensor is on the tag, this is set as TRUE.

DSFID-Pad-Bytes: [Optional] INTEGER
This is the number of additional bytes requested by application to support additional arguments to be added at a later time, e.g. the **Memory-Capacity** and/or the **Length-Of-Encoded-Data**. In a response this might also include the number of pad bytes added to enable the extended DSFID to be locked on a lock-block boundary.

Memory-Capacity: INTEGER
This is only included in a response that included the Ext-DSFID-Constructs argument

Length-Of-Encoded-Data: INTEGER
This is only included in a response that included the Ext-DSFID-Constructs argument

ISO-UIImemory ۵-۱۱

ISO-UIImemory argument

Protocol-Control-Word: BYTE STRING (2)

DSFID: BYTE [OPTIONAL]
This is not returned when the AFI declares that that the Object is a Monomorphic-UII

Read-Objects-Response-List: List of <Read-Objects-Response>

Item-Related-Add-Objects ۶-۱۱

متغیر فرمان **Item-Related-Add-Objects**، کارکرد و ساختاری مشابه متغیر فرمان **Add-Objects** را دارد (به زیربند ۱۱-۱ مراجعه کنید)، به جز اینکه در قطعه مربوط به عنوان یک برچسب RFID کدبندی می‌شود که می‌تواند چند قطعه را در تراکنش‌ها واسط هوایی همانند مشخص کند. این متغیر ویژه، سبب می‌شود که از **UII-Add-Objects** تشخیص داده شود (به زیربند ۱۱-۱۶ مراجعه کنید).

Item-Related-DSFID-Constructs ۷-۱۱

متغیر فرمان **Item-Related-DSFID-Constructs**، کارکرد و ساختاری مشابه متغیر فرمان **DSFID-Constructs** دارد (به زیربند ۱۱-۲ مراجعه کنید)، به استثنای اینکه در اقلام مربوط به عنوان یک برچسب RFID کدبندی می‌شود که می‌تواند چند اقلام را در تراکنش‌ها واسط هوایی همانند مشخص کند. . این متغیر ویژه، سبب می‌شود که از **UII-DSFID-Constructs** تشخیص داده شود (به زیربند ۱۱-۱۷ مراجعه کنید).

Multiple-Records-Constructs ۸-۱۱

این متغیر، فهرستی از متغیرهای اضافی را فراهم می‌کند و هنگامی استفاده می‌شوند که یک رکورد چند تایی در Logical Memory نوشته می‌شود.

متغیر فرمان **Append-To-Existing-Multiple-Record**، که از نووع BOOLEAN است(به زیربند ۷-۴-۴ مراجعه کنید)، استفاده می‌شود تا اعلان کند، اگر TRUE است، خواه فهرست اشیا داده به رکورد قبلی افزوده شود، تحت تاثیر همه فیلدهای تطابق متغیرهای مولفه که برای رکورد چندتایی کدبندی شده است، می‌باشد. اگر FALSE باشد، این فرمان باید برای ساخت یک رکورد جدید استفاده شود.

متغیر فرمان **Application-Defined-Record-Capacity** (به زیربند ۷-۴-۵ مراجعه کنید) به آسانی یک کاربرد انتخابی را فراهم می‌کند تا ظرفیت حافظه را در یک اندازه معین تنظیم کند و یا به Data Processor اجازه دهد تا ردیف رکورد را بلوک‌بندی کند. اجازه دادن به Data Processor برای کنترل آن توصیه می‌شود مگر اینکه یک متغیری برای برنامه کاربردی وجود داشته باشد تا یک ظرفیت حافظه بیشتر را قرار دهند، برای

مثال، اقلام های داده‌های اضافی بتوانند در یک زمان کدبندی شوند. بنابراین **Record-Memory-Capacity** تنها زمانی لازم است که برنامه کاربردی باید این شکل از حافظه را کنترل کند.

متغیر **Record-Type-Classification** (به زیربند ۷-۴-۵۶ مراجعه کنید) باید مطابق با قواعد تعریف شده در MR-header باشد. اگر مقدار نادرستی استفاده شود، این فرمان باید با Data Processor رد شود.

زمانی که یک فرمان ساخته می‌شود تا یک رکوردی را که بخشی از یک ساختار سلسله مراتبی می‌باشد، کدبندی کند، مرجع باید برای استاندارد کاربردی ساخته شود تا اطمینان دهد که ارتباطات سلسله مراتبی صحیح هستند. متغیر **Identifier-of-My-Parent** (به زیربند ۷-۴-۲۲ مراجعه کنید) باید براساس یک رکوردی باشد که از قبل موضوع یک فرمان نوشتن قبلی بوده است. اگر یک عدم انطباق وجود داشته باشد، آنگاه این رکورد کدبندی نمی‌شود.

متغیر **Number-In-Data-Element-List** (به زیربند ۷-۴-۴۱ مراجعه کنید) تنها زمانی به کار برده می‌شود که رکورد، یک فهرست اقلام داده است و اطلاعات اضافی را به Data Processor می‌دهد تا مطمئن شود که تعداد صحیح اقلام های داده با همان **Relative-OID** صحیح کدبندی می‌شود. این ممکن است به خصوص زمانی مفید باشد که داده‌ها در قالب و طول بین نمونه‌های اقلام داده متفاوت می‌باشند.

متغیرهای **Lock-Record-Preamble** (به زیربند ۷-۴-۳۰ مراجعه کنید)، **Update-Multiple-Records-Directory** (به زیربند ۷-۴-۶۲ مراجعه کنید) و **Lock-Directory-Entry** (به زیربند ۷-۴-۲۸ مراجعه کنید) دستورالعمل‌هایی را به Data Processor می‌دهند و بطور کلی به رکورد موضوع، مربوط می‌شوند. بنابراین اگر متغیر **Update-Multiple-Records-Directory** به TRUE تنظیم شود و هنوز فهرستی وجود نداشته باشد، باید یک فهرست کامل ساخته شود. صرف‌نظر از موقعیت متغیر، اگر یک فهرست کدبندی شده باشد، Data Processor باید به‌طور خودکار فهرست را به‌طورکامل به‌روزرسانی کند.

Multiple-Records-Constructs argument

Append-To-Existing-Multiple-Record BOOLEAN

If set to TRUE, the Data Processor shall check that all the arguments match with an existing multiple record and that none of the **Relative-OID** values already exist in the current version of the record. In the case that the existing record is defined as a data element list, then none of the list element number values in this command shall already be encoded in the current version of the record. If set to FALSE, the Data Processor shall encode the record as a new record.

Application-Defined-Record-Capacity: BOOLEAN

If set as TRUE, the application needs to define the Record-Memory-Capacity argument. If FALSE, the Data Processor shall simply block align the record and encode the record capacity as the minimum required to support the record.

Record-Memory-Capacity: [Optional] INTEGER

This is the total number of blocks that the application requires for the record, with any unused blocks to be set aside for additional encoding in this record in the future.

Record-Type-Classification: BIT STRING

Possible Values:

Value	Definition
000	stand-alone record, with an instance-of arc = 0
001	stand-alone record, with an instance-of arc >0
010	hierarchical record, top level
011	hierarchical record, has both parent and child(ren)
100	hierarchical record, data element list
101	other hierarchical record, no further children
110	Not relevant to this command (because it is associated with deleted records)
111	reserved

Identifier-Of-My-Parent: [Conditional] INTERGER

This value is the same as the hierarchical identifier of the parent record

Number-In-Data-Element-List: [Conditional] INTERGER (1..255)

This value is provided when the record classification is {100} to assist the data processor in encoding the associated Add-Objects argument

Lock-Record-Preamble: BOOLEAN

If this is set as TRUE, then the preamble shall be block aligned and locked

Update-Multiple-Records-Directory: BOOLEAN

If this is set as TRUE, then the directory is updated with this record, and any others not on the directory

Lock-Directory-Entry: BOOLEAN

If this is set as TRUE, then the directory entry for the record shall be block aligned and locked

۱۱-۹ Multiple-Records-Directory-Structure

این متغیر پاسخ، فهرستی از محتوای چندین فهرست رکوردها تهیه می‌کند بطوری‌که اطلاعات می‌توانند توسط برنامه کاربردی استفاده شوند تا فرمآن‌های اضافی را بسازند.

متغیر **Multiple-Records-Directory-Structure** شامل متغیرها و نام‌های فیلد زیر می‌باشد:

DSFID-Constructs (به زیربند ۱۱-۲ مراجعه کنید)

Ext-DSFID-Constructs (به زیربند ۱۱-۴ مراجعه کنید)

Hierarchical-Identifier-Arc (به زیربند ۷-۴-۲۱ مراجعه کنید)

Identifier-of-My-Parent (به زیربند ۷-۴-۲۲ مراجعه کنید)

Instance-of-Arc (به زیربند ۷-۴-۲۵ مراجعه کنید)

Record-Type-Arc (به زیربند ۷-۴-۵۵ مراجعه کنید)

Record-Type-Classification (به زیربند ۷-۴-۵۶ مراجعه کنید)

Sector-Identifier (به زیربند ۷-۴-۵۷ مراجعه کنید)

Start-Address-of-Record (به زیربند ۷-۴-۵۹ مراجعه کنید)

Multiple-Records-Directory-Structure argument

DSFID-Constructs-list: [Conditional] List of <DSFID-Constructs>

Ext-DSFID-Constructs-list: [Conditional] List of <Ext-DSFID-Constructs>

Sector-Identifier: [Conditional] INTEGER

If the sector identifier has not been provided by the MR-header, then the value in the record preamble is valid

Record-Type-Classification: BIT STRING

Possible Values:

Value	Definition
000	stand-alone record, with an instance-of arc = 0
001	stand-alone record, with an instance-of arc >0
010	hierarchical record, top level
011	hierarchical record, has both parent and child(ren)
100	hierarchical record, data element list
101	other hierarchical record, no further children
110	Not relevant to this command (because it is associated with deleted records)
111	reserved

Record-Type-Arc: INTEGER

Instance-Of-Arc: [Conditional] INTEGER

Hierarchical-Identifier-Arc: [Conditional]

INTEGER **Identifier-Of-My-Parent**: [Conditional]

INTERGER

This value is the same as the hierarchical identifier of the parent record

Start-Address-Of-Record: EBV-8

۱۰-۱۱ Multiple-Records-Header-Structure

این متغیر پاسخ، فهرستی از محتوای MR-header را به‌گونه‌ای فراهم می‌کند که اطلاعات را می‌توان توسط برنامه کاربردی استفاده کرد تا فرمان اضافی ساخته شوند.

متغیر **Multiple-Records-Header-Structure** شامل متغیرها و نام‌های فیلد زیر می‌باشد.

DSFID-Constructs (به زیربند ۱۱-۲ مراجعه کنید)

Ext-DSFID-Constructs (به زیربند ۱۱-۴ مراجعه کنید)

Multiple-Records-Header-structure (به زیربند ۷-۴-۳۸ مراجعه کنید)

Multiple-Records-Features-Indicator (به زیربند ۷-۴-۳۹ مراجعه کنید)

Number-of-Records (به زیربند ۷-۴-۴۲ مراجعه کنید)

Pointer-To-Multiple-Records-Directory (به زیربند ۷-۴-۵۱ مراجعه کنید)

Sector-Identifier (به زیربند ۷-۴-۵۷ مراجعه کنید)

Multiple-Records-Header-Structure argument

DSFID-Constructs-list: List of <DSFID-Constructs>

Ext-DSFID-Constructs-list: List of <Ext-DSFID-Constructs>

Multiple-Records-Directory-Length: [Optional] EBV-8

Multiple-Records-Features-Indicator: BYTE
This is a bit map that is set in this command that determines rules for the Data Processor to follow when encoding individual records.

Sector-Identifier: INTEGER
Possible Values:

Value	Definition
0	Indicates that the sector identifier varies between records, and that the true value is only obtainable for the individual record.
≠0	Indicates the true value of the sector that applies to all records.

Pointer-To-Multiple-Records-Directory: EBV-8
This value is based on responses from the interrogator on the memory mapping of the RFID tag that is being addressed.

Number-Of-Records: EBV-8
If bit 2 of the **Multiple-Records-Features-Indicator** = 0, then the value of this field is probably invalid and shall be ignored.

۱۱-۱۱ Multiple-Records-Preamble-Structure

این متغیر پاسخ، فهرستی از محتوای مقدمه رکورد را به‌گونه‌ای فراهم می‌کند که اطلاعات را می‌توان توسط برنامه کاربردی استفاده کرد تا فرمان‌های اضافی ساخته شوند.

متغیر **Multiple-Records-Preamble-Structure** شامل متغیرها و نام‌های فیلد زیر است:

Data-Length-of-Record (به زیربند ۷-۴-۱۲ مراجعه کنید)

DSFID-Constructs (به زیربند ۱۱-۲ مراجعه کنید)

Encoded-Memory-Capacity (به زیربند ۷-۴-۱۸ مراجعه کنید)

Ext-DSFID-Constructs (به زیربند ۱۱-۴ مراجعه کنید)

Hierarchical-Identifier-Arc (به زیربند ۷-۴-۲۱ مراجعه کنید)

Identifier-Of-My-Parent (به زیربند ۷-۴-۲۲ مراجعه کنید)

Instance-Of-Arc (به زیربند ۷-۴-۲۵ مراجعه کنید)

Record-Type-Arc (به زیربند ۷-۴-۵۵ مراجعه کنید)

Record-Type-Classification (به زیربند ۷-۴-۵۶ مراجعه کنید)

Sector-Identifier (به زیربند ۷-۴-۵۷ مراجعه کنید)

Multiple-Records-Preamble-Structure argument

DSFID-Constructs-list: List of <DSFID-Constructs>

Ext-DSFID-Constructs-list: [Conditional] List of <Ext-DSFID-Constructs>

Encoded-Memory-Capacity: EBV-8
This is the size of the memory that has been reserved for the record, in terms of write blocks.

Data-Length-Of-Record: [Conditional] EBV-8
This is the size of the encoded record, in terms of write blocks.

Sector-Identifier: [Conditional] INTEGER
If the sector identifier has not been provided by the MR-header, then the value in the record preamble is valid

Record-Type-Classification: BIT STRING
Possible Values:

Value	Definition
000	stand-alone record, with an instance-of arc = 0
001	stand-alone record, with an instance-of arc >0
010	hierarchical record, top level
011	hierarchical record, has both parent and child(ren)
100	hierarchical record, data element list
101	other hierarchical record, no further children
110	Not relevant to this command (because it is associated with deleted records)
111	reserved

Record-Type-Arc: INTEGER

Instance-Of-Arc: [Conditional] INTEGER

Hierarchical-Identifier-Arc: [Conditional] INTEGER

Identifier-Of-My-Parent: [Conditional] INTERGER
This value is the same as the hierarchical identifier of the parent record

۱۲-۱۱ Packed-Objects-Constructs

متغیرهای زیر تنها برای **Packed-Objects** به کار برده می‌شوند و اگر **Access-Method**، **Packed-Objects** نباشد، نادیده گرفته می‌شوند.

متغیر **Packed-Objects-Constructs** متغیرهای زیر را دارد:

Block-Align-Packed-Objects (به زیربند ۷-۴-۹ مراجعه کنید)

Editable-Pointer-Size (به زیربند ۷-۴-۱۷ مراجعه کنید)

ID-Type (به زیربند ۷-۴-۲۴ مراجعه کنید)

Object-off sets-Multiplier (به زیربند ۷-۴-۴۴ مراجعه کنید)

Packed-Object-Directory-Type (به زیربند ۷-۴-۴۵ مراجعه کنید)

PO-Directory-Size (به زیربند ۷-۴-۴۸ مراجعه کنید)

PO-ID-Table (به زیربند ۷-۵-۱۰ مراجعه کنید)

PO-Index-Length (به زیربند ۷-۴-۴۹ مراجعه کنید)

Packed-Objects-Constructs argument

PO-ID-Table: OCTET STRING

ID-Type: INTEGER (0..15)

Possible Values:

Value	Definition (see 7.4.24 for further details)
0	ID List
1	ID Map
2 to 15	Reserved for future definition

Packed-Object-Directory-Type: INTEGER (0..15)

Possible Values:

Value	Definition (see 7.4.45 for further details)
0	This **Packed-Object** is not a directory and does not require one
1	**Packed-Object** Presence/Absence
2	**Packed-Object** index field
3	**Packed-Object** offset
4	**Packed-Object** pointer-allocation only expecting a future command to set the directory type
5 to 15	Reserved for future definition

PO-Index-Length: INTEGER (1…7)

If this parameter is present and **Packed-Object-Directory-Type** is 2 **(Packed-Object** index field) then the implementation shall use this parameter in the POIndex Length parameter created for the POIndex Field for this Packed Object in a PO index directory. If the **Directory-Type** is not 2, this parameter shall be ignored

Object-Offsets-Multiplier: INTEGER

If this parameter is present and **Packed-Object-Directory-Type** is 3 **(Packed-Object** offset) and **PO- Index-Length** is present, then the implementation shall use this parameter to reserve the number of bits of storage for object offsets in an AuxMap section of the directory Packed Object. If the **Directory- Type** is not 3 or a **PO-Index-Length** parameter is not present, this parameter shall be ignored. If the implementation is not able to allocate the input size number of bits for the AuxMap section of the directory Packed Object, the implementation shall return an **Insufficient-Tag-Memory** completion code.

PO-Directory-Size: INTEGER

If this parameter is present and **Packed-Object-Directory-Type** is 4 **(Packed-Object** pointer allocation only expecting a future command to set the directory type) then the implementation shall use this parameter for appropriate sizing of the (null) directory pointer created for this Packed-Object. If the **Directory-Type** is not 4, this parameter shall be ignored. If the implementation is not able to allocate the input size number of bits for the Addendum Packed Object, the implementation shall return an **Insufficient-Tag-Memory** completion code.

Block-Align-Packed-Objects: BOOLEAN

If set to TRUE, the interrogator shall ensure that this **Packed-Object** begins on a block boundary and that any necessary pad bytes are added after a previous **Packed-Object** (if present).

Editable-Pointer-Size: INTEGER

If set to a non-zero value, the interrogator shall mark the Packed Object as editable and create a pointer to an Addendum Packed Object of the size of the parameter in bits. If set to zero it indicates that the addendum subsection shall not be included.

Read-Objects ۱۳–۱۱

Read-Objects argument

Object-Identifier: OBJECT IDENTIFIER

Check-Duplicate: BOOLEAN
If set to TRUE, the interrogator shall check that there is only one occurrence of the OBJECT IDENTIFIER encoded on the RFID tag.

Read-Objects-Response ۱۴–۱۱

Read-Objects-Response argument

Object-Identifier: OBJECT IDENTIFIER

Object: BYTE STRING

Compact-Parameter: INTEGER
Possible Values:

Value	Definition
0	Application-Defined
2	UTF8-Data
14	De-Compacted-Monomorphic-UII
15	De-Compacted-Data

Lock-Status: BOOLEAN
If TRUE, the Data-Set or Packed-Object containing the Object Identifier and Object is locked.

Completion-Code: INTEGER
These definitions are supplementary to those of the response that includes this argument
Possible Values:

Value	Definition
0	No-Error
10	Duplicate-Object
13	Object-Identifier-Not-found
15	Object-Not-Read
35	Monomorphic-UII-OID-Mismatch

Read-OIDs-Response ۱۵–۱۱

Read-OIDs-Response argument

Object-Identifier: OBJECT IDENTIFIER

UII-Add-Objects –۱۶–۱۱

متغیر فرمان **UII-Add-Objects** عملکرد و ساختاری همانند متغیر فرمان **Add-Objects** دارد (به زیربند ۱۱-۱ مراجعه کنید)، به استثنای اینکه در قطعه UII یک برچسب RFID کدبندی می‌شود که می‌تواند چند قطعه را در تراکنش‌های واسط هوایی مشابه مشخص کند. این متغیر باید شامل یک **Object-Identifier** و **Object** باشد. این متغیر ویژه، سبب می‌شود که از **Item-Related-Add-Objects** تشخیص داده شود (به زیربند ۱۱-۶ مراجعه کنید).

UII–DSFID-Constructs ۱۷–۱۱

متغیر فرمان **UII–DSFID-Constructs** عملکرد و ساختاری همانند متغیر فرمان **DSFID-Constructs** دارد (به زیربند ۱۱-۲ مراجعه کنید) به استثنای اینکه در قطعه UII یک برچسب RFID کد می‌شود که می‌تواند چند قطعه را در تراکنش‌های و واسط هوایی همانند مشخص کند. . این متغیر ویژه، سبب می‌شود که از **Item-Related-DSFID-Constructs** تشخیص داده شود (به زیربند ۱۱-۷ مراجعه کنید).

Write-Responses ۱۸–۱۱

Write-Responses argument

Object-Identifier: OBJECT IDENTIFIER

Completion-Code: INTEGER
These definitions are supplementary to those of the response that includes this argument

Possible Values:

Value	Definition
0	No-Error
9	Object-Not-Added
10	Duplicate-Object
11	Object-Added-But-Not-Locked

پیوست الف
(اطلاعاتی)
قواعد نحوی انتزاعی و قواعد انتقال کدبندی

الف-۱ قواعد نحوی انتزاعی

این پیوست قواعد نحوی انتزاعی را نشان می‌دهد که در استاندارد ISO/IEC 15961: 2004 به کار برده شده است تاپیاده‌سازی پروتکل داده‌ها را براساس آن استاندارد و استاندارد ISO/IEC 15962: 2004 پشتیبانی کند. پیوست ث از این استاندارد، ۱۶ فرمان اصلی را با استفاده از قواعد نحوی انتزاعی استاندارد ISO/IEC 15961: 2004 نشان می‌دهد. اینها را می‌توان به عنوان مرجع استفاده کرد تا با سبک موجود در فرمان‌های کارکردی مقایسه شوند.

قواعد نحوی انتزاعی تعیین شده در استاندارد ISO/IEC 15962: 2004 براساس ASN.1 می‌باشد همان‌طور که در استاندارد ISO/IEC 8824-1 تعریف شده است. این علامت‌گذاری باید در آن استاندارد مشخص شود.

الف-۱-۱ ترتیب نویسه

ترتیب نویسه استفاده شده است تا یک عنوان ASN.1 را تعریف کنید که باید شامل اینها باشد:

A تا Z

a تا z

0 تا ۹

! ; * ^ & □ " ' - [ ] ( ) @ . < } { , = :

این ترتیب نویسه یا علامت همانند نویسه تعریف شده در استاندارد ISO/IEC 8824-1 می‌باشد.

الف-۱-۲ انواع فراگیر[1]

ASN.1 از تعدادی انواع فراگیر پشتیبانی می‌کند که در قواعد نحوی، اساسی هستند، گاهی اوقات «انواع درون‌ساخت[2]» نامیده می‌شوند. هر کدام، یک برچسب کلاس را در استاندارد ISO/IEC 8824-1 تخصیص داده‌اند تا بدون ابهام هر نوع داده‌ای را تشخیص دهند. انواع فراگیر در حروف بزرگ نشان داده می‌شوند، مثلا **UNIVERSAL**.

Universal Types در استاندارد ISO/IEC 15962: 2004 همراه با Class Tags آن‌ها استفاده شده‌اند که در جدول پیوست الف-۱، Universal Types استفاده شده در استاندارد ISO/IEC 15962: 2004 ، نشان داده می‌شوند.

1 - Universal Types
2 - Built-in Types

جدول الف-۱- Universal Types استفاده شده در استاندارد ISO/IEC 15962: 2004

برچسب کلاس	نوع فراگیر
۱	BOOLEAN
۲	INTEGER
۶	OBJECT IDENTIFIER
۴	OCTET STRING
۱۳	RELATIVE-OID (reserved for future commands)
۱۶	SEQUENCE & SEQUENCE OF

الف-۱-۳ مراجع نوع

علاوه بر Universal Types، ASN.1 انواع ویژه برنامه کاربردی را می‌تواند تعریف کند. زمانی که یک نوع تعریف می‌شود، یک نام ارائه داده می‌شود تا در تخصیص نوع دیگر، به آن ارجاع دهد. مرجع‌های Type با یک حرف بزرگ شروع می‌شوند و نام کامل بدون فضای خالی نشان داده می‌شوند. چند متغیر برای ارائه نویسههای بعدی وجود دارد. استاندارد ISO/IEC 15962: 2004 از قرارداد ترکیبی نویسههای بزرگ و کوچک استفاده می‌کند.

مثال: **TypeReference**.

نام TypeReference توسط سه نویسه "=::" دنبال می‌شود تا آن را از تعریف آن جدا کنند.

مثال‌های نام‌های TypeReference که در استاندارد ISO/IEC 15962: 2004 استفاده می‌شوند اینها هستند:

ApplicationFamilyId
ObjectId
StorageFormat
TagId

تمام نام‌های TypeReference به کار برده شده در استاندارد ISO/IEC 15962: 2004، در زیربند مناسبی از این پیوست تعریف می‌شوند.

الف-۱-۴ نام‌های اقلام

مولفه‌ها یا اقلام یک TypeReference یا فهرست شمارشی با استفاده از یک حرف کوچک در شروع، نام‌گذاری می‌گردد، مثال، **elementName**. برای برخی اقلام ، نمونه نوعی دیگر لازم است که در یک TypeReference یا یک Universal Type باشد.

مثال‌های elementNames که در استاندارد ISO/IEC 15962: 2004 استفاده می‌شوند اینها هستند:

accessMethod
applicationFamilyId
applicationsubFamily
commandCode

compactParameter
object
objectld
tagld

یادآوری- در استاندارد ISO/IEC 15962: 2004، چند نام **elementNames** و **TypeReference** اغلب همانند هستند به استثنای اینکه اولین حرف برای **element Name**، حرف کوچک و برای **TypeReference**، حرف بزرگ می‌باشد.

تمام **elementNames** به کار برده شده در استاندارد ISO/IEC 15962: 2004 در زیربندهای مناسب تعریف می‌شوند.

الف-۱-۵ سایر قراردادهای ASN.1 شرح داده شده

با استفاده از یک مثال ساده‌ای از قواعد نحوی ASN.1، نامربوط با هدف، می‌توان ویژگی‌های قواعد نحوی را شرح داد.

مثال:

```
CustomerOrder : : = SEQUENCE {
    orderNumber    INTEGER
    name           OCTET
    STRING                 address
                   CustomerAddre
    ss  productDetails  SEQUENCE
    OF SEQUENCE {
                   productCode    OBJECT IDENTIFIER
                   quantity       INTEGER (1..999)
                   } ,
    urgency        ENUMERATED {
                   nextDay (0),
                      -- excludes Saturday and Sunday
                      firstClass (1),
                      roadTransport (2),
                      -- typically three days
                      }
}
```

در این مثال:

- علامت "::=" ، **Customerorder** (که یک TypeReference است) نام‌گذاری شده را از تعریف جدا می‌کند.
- "{}" که یک **SEQUENCE** و پایان را دنبال می‌کند، تعیین می‌کنند که تمام **orderNumber**، **name**، **address**، **ProductDetails** و **Urgency**، اقلامی از مرجع نوع CustomerOrder می‌باشند.

- نام اقلام **address** در مرجع نوع **CustomerAddress** بیشتر مشخص می‌شود (در این مورد، برای اختصار، مثال زده نمی‌شود).

- اقلام **productDetails** شامل دو اقلام دیگر **productCode** و **quantity** می‌باشد. این جفت اقلام n بار تکرار می‌شود که براساس **SEQUENCE OF SEQUENCE** می‌باشد. { }، محدده Type را مشخص می‌کند.

- اقلام **Urgency** یکی از سه کد: ۰ و ۱ و یا ۲ را توسط استفاده از نوع **ENUMERATED** نشان می‌دهد. { }، محدده Type را مشخص می‌کند.

- توضیحات در این مثال، مانند "exclude Saturday and Sunday" و "typically three days"، در جلوی دو خط تیره "—" قرار می‌گیرند.

- مقدار **INTEGER** برای اقلام **quantity** که در محدوده "(1..999)" است، هر مقداری در دامنه ۱ تا ۹۹۹ می باشد و اگر مقدار ۱۰۰۰ یا اقلام بیشتر از **ProductCode** سفارش داده شود، خطا ایجاد می‌شود.

الف-۱-۶ ساختار پودمانی قواعد نحوی ASN.1

با حفظ استانداردهای ASN.1، قواعد نحوی در استاندارد ISO/IEC 15962: 2004 در یک قالب پودمانی ارائه می‌شود. پودمان‌های جدا برای فرمان‌ها و برای پاسخ‌ها استفاده می‌شوند. هر پودمان شامل موارد زیر است:

- یک نام انحصاری

- یک Object-Identifier انحصاری که به این استاندارد ویژه اشاره می‌کند (مطابق با استاندارد ISO/IEC 8824-1) قوس یکی مانده به آخر "Command Modules (126)" یا "response modules (127)" می‌باشد تا جریان داده‌ها را تشخیص دهد. قوس نهایی هر جفت فرمان/پاسخ، نام و مقدار مشابهی دارد تا اینها را با هم پیوند دهد[1].

- استفاده از کلمات کلیدی DEFINITONS، BEGIN و END مطابق با استاندارد ISO/IEC 8824-1 می‌باشد و این پودمان‌ها را می‌توان از طریق ابزارهای کامپایلر پردازش کرد.

- جمله‌ی[2] "EXPLICIT TAGS"، که این استاندارد استفاده می‌کند، نشان می‌دهد تمام اقلام در نهایت با عنوان UNIVERSAL TYPES کدبندی شده‌اند.

ساختار یک پودمان فرمان طبق قالب مشترک زیر می‌باشد:

```
Module Name
    {ISO(1) standard(0) rfid-data-protocol (15961) commandModules (126)
                                moduleName(n)}
  DEFINITIONS
  EXPLICIT TAGS ::=
  BEGIN
```

1 - Link
2 - Statement

```
CommandName
-- assignments

END
```

responseModules یک قالب مشابهی را دنبال می‌کند.

در هر پودمان، تمام اقلام طوری تعریف می‌شوند که اینها را در UNIVERSAL TYPES کاهش می‌دهند. نیازی به اجرای هر تابع ورودی در پودمان نمی‌باشد.

الف-۲ قواعد نحوی انتقال

الف-۲-۱ ساختار کدبندی انتقال

ساختار کدبندی انتقال که برای پروتکل داده برای RFID برای مدیریت اقلام می‌باشد (همان‌طور که در اصل در استاندارد ISO/IEC 15961: 2004 مشخص شده است)، در قسمت زیر شرح داده می‌شود:

۱. هشتایی‌هایی شناسه نوع که برچسب ASN.1 (کلاس و عدد) از Type را کدبندی می‌کنند برای توصیف مقدار داده استفاده شده‌اند؛
۲. بایت‌های طول که تعداد بایت‌هایی که محتواها را می‌سازند، را تعریف می‌کنند ؛
۳. بایت‌های محتوا (یا مقدار).

گاهی اوقات با عنوان Type، Length، Value (**TLV**) شناخته شده است. زمانی که کدبندی براساس یک ترتیبی از **TLV** می‌باشد، به عنوان کدبندی اولیه شناخته شده است.

مقدار V می‌تواند سه‌تایی[1] **TLV** باشد و زمانی که از این ساختار استفاده می‌شود با عنوان کدبندی ساختاریافته شناخته شده است. برای مثال، **TL TLV TLV TLV**. انتخاب بین کدبندی اولیه و ساختاریافته بیشتر با قواعد کدبندی اصلی استاندارد ISO/IEC 8825-1 تعیین می‌شود.

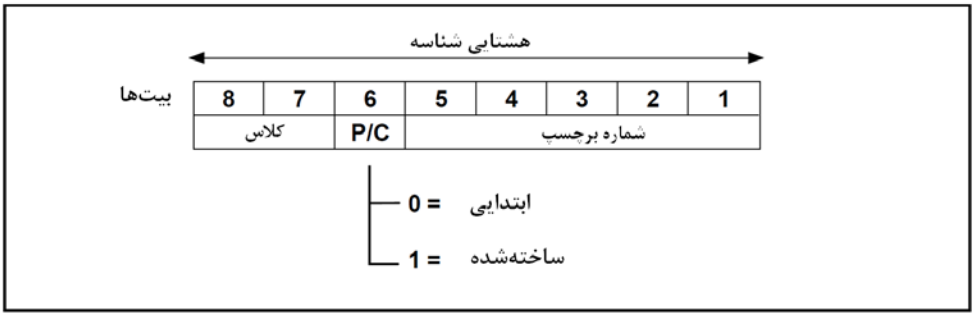
انواع سازنده SEQUENCE و SEQUENCE OF باید از یک کدبندی ساختار یافته استفاده کنند تا به‌طورکامل مطابق با استاندارد ASN.1 باشند. در غیراین‌صورت قواعد استاندارد ISO/IEC 8825-1 لازم است که از کدبندی Primitive استفاده کند و یا یک انتخاب را پیشنهاد کند که در آن حالت تنها کدبندی اولیه در استاندارد ISO/IEC 15961: 2004 استفاده می‌شوند. این انتخاب برای هر یک از Universal Types بطور واضح تعریف می‌شود که قواعد کدبندی BER خودشان را دارند که در بندهای بعدی تعریف می‌شوند.

پودمان OBJECT IDENTIFIER باید در یک ساختار TLV در شروع انتقال کدبندی شود.

الف-۲-۲ کدبندی شناسه نوع ASN.1

شناسه نوع ASN.1 باید به عنوان یک هشتایی برای Types تعریف شده در استاندارد ISO/IEC 15961:2004 کدبندی شود، همان‌طور که در شکل الف-۱- ساختار هشتایی شناسه نوع - شرح داده شده است.

1 - Triple



که بیت‌های ۸ و ۷ کلاس برچسب ASN.1 را کدبندی می‌کنند.
بیت ۶ کدبندی می‌کند که آیا این یک برچسب اولیه (Primitive) است یا برچسب ساخته‌شده (Constructed)
بیت‌های ۵ تا ۱ تعداد برچسب ASN.1 را کدبندی می‌کنند.

شکل الف-۱- ساختار هشتایی شناسه نوع

مقدار دو بیت برای Class باید یکی از مقادیر تعریف شده در جدول الف-۲- کدبندی کلاس ASN.1 از برچسب- می‌باشد.

جدول الف-۲- کدبندی کلاس ASN.1 از برچسب

کلاس	بیت ۸	بیت ۷
Universal	۰	۰
Application	۰	۱
Context-specific	۱	۰
Private	۱	۱

مقدار تک بیت برای مولفه "P/C" باید به "0_2" تنظیم شود تا ساختارهای کدبندی اولیه را نشان دهد و یا باید به "1_2" تنظیم شود تا ساختارهای کدبندی ساخته‌شده را نشان دهند.
مقدار ۵بیت برای برچسب ASN.1 باید تعداد برچسب Class را به صورت یک عدد صحیح دودویی با بیت ۵ به‌عنوان پرارزش‌ترین بیت، کدبندی کند. برچسب‌های Class تعیین شده برای استاندارد ISO/IEC 15961:2004 در جدول الف-۱- Universal Types استفاده شده در استاندارد ISO/IEC 15961:2004 - تعریف می‌شوند

مثال:

Universal Type = OBJECT IDENTIFIER
ASN.1 Type identifier = 00 0 00110

الف-۲-۳ کدبندی طول

اگرچه استاندارد ISO/IEC 8825-1 شکل‌های دیگری از کدبندی طول را امکان‌پذیر می‌کند، تنها شکل معین[1] باید در استاندارد ISO/IEC 15961:2004 استفاده شود زیرا هم در کدبندی اولیه و هم در کدبندی ساخته‌شده به کاربرده می‌شود. برای شکل معین، بایت‌های طول باید شامل یک یا چند بایت باشند و به تعداد بایت‌ها در محتواها بستگی دارد. اگر تعداد بایت‌ها در محتواها کمتر یا برابر با ۱۲۷ باشد، سپس یک هشتایی طول واحد باید استفاده شود. بیت ۸ باید "0_2" و بیت‌های ۷ تا ۱ باید تعداد بایت‌ها را در محتوا کدبندی کنند (که ممکن است صفر باشد) همانند یک عدد صحیح دودویی بدون‌علامت[2] با بیت ۷ به‌عنوان پرارزش‌ترین بیت.

مثال:

L = 38 is encoded as 00100110_2

اگر تعداد بایت‌ها در محتواها بیشتر از ۱۲۷ باشد، آنگاه دو یا چند بایت طول، باید استفاده شود. این طول باید به یک مقدار ردیفی هشتایی[3] تبدیل شود، برای مثال، یک طول ۲۰۱ بایت، به $C9_{16}$ (یا 11001001) تبدیل می‌شود. این مقدار در بایت‌های دوم و بعدی کدبندی می‌شود. اولین هشتایی باید این‌طور کدبندی شود:

الف- بیت ۸ باید 1_2 باشد.

ب- بیت‌های ۷ تا ۱ باید تعداد بایت‌های بعدی را در بایت‌های طول کدبندی کنند، همانند یک عدد صحیح دودویی بدون علامت با بیت ۷ به‌عنوان پرارزش‌ترین بیت.

ج- مقدار 11111111_2 نباید برای تعمیم کلی زیر استفاده شود:

مثال

Length of content = 357
Convert to HEX = 01 65_{16}
= 00000001_2 01100101_2
As this is 2 bytes, the first octet = 100000010_2
The complete length encoding is:
10000010 00000001 01100101_2
= 82 01 65_{16}

الف-۲-۴ هشتایی‌های محتواها

هشتایی‌های محتواها، مقدار داده‌ها را کدبندی می‌کنند که می‌تواند صفر باشد یا یک یا چند هشتایی باشد که به Universal Type بستگی دارد همان‌طور که در زیربندهای بعدی تعیین شده است.

الف-۲-۵ کدبندی یک مقدار BOOLEAN

کدبندی یک مقدار BOOLEAN باید اولیه باشد تا مطابق با استاندارد ISO/IEC 8825-1 باشد. مقدار BOOLEAN باید در یک هشتایی کدبندی شود. اگر مقدار BOOLEAN، FALSE باشد، هشتایی باید صفر

1 - Definite Form
2 - Unsigned Binary Integer
3 - Octet Align Value

باشد. اگر مقدار BOOLEAN، TRUE باشد، این هشتایی باید مقداری غیرصفر داشته باشد که در اختیار فرستنده است.

الف-۲-۶ کدبندی یک مقدار INTEGER

کدبندی یک مقدار INTEGER باید اولیه باشد تا مطابق با استاندارد ISO/IEC 8825-1 باشد. این عدد صحیح باید در یک یا چند هشتایی با استفاده از رویه زیر کدبندی شود.

برای اعداد صحیح مثبت و صفر:

۱- عدد کلی به یک عدد صحیح دودویی در یک فیلد بیت تبدیل می‌شود که پرارزش‌ترین بیت، اولین است.

۲- فیلد بیت در حدود هشتایی با افزودن بیت‌های صفر مقدم، ردیف می‌شود.

۳- اگر بیت مرتبه بالا ۱ باشد، یک هشتایی فضای خالی 00_{16} را به عنوان یک پیشوند اضافه کنید.

یادآوری- بیت مرتبه بالا با مقدار 0 نشان دهنده این است که این کدبندی از یک مقدار عدد صحیح مثبت است.

مثال:

Integer 128
Step 1: 10000000
Step 2: 10000000
Step 3: 00000000 10000000

برای اعداد صحیح منفی، این کدبندی در یک قانون مکمل- دوها[1] می‌باشد.

۱- عدد به یک عدد صحیح دودویی در یک فیلد تبدیل می‌شود که پرارزش‌ترین بیت، اولین است.

۲- فیلد بیت در حدود هشتایی با افزودن بیت‌های صفر مقدم، ردیف می‌شود.

۳- مقدار دودویی از مرحله ۲، مکمل بیت شده است (یعنی ۰ به ۱، ۱ به ۰).

۴- قانون مکمل- دوها با افزودن 1_2، به رشته بیت مرحله ۲ به کار برده می‌شود.

۵- اگر بیت مرتبه بالا صفر باشد، یک هشتایی فضای خالی FF_{16}، را به عنوان یک پیشوند اضافه کنید.

یادآوری- بیت مرتبه بالا با مقدار ۱، نشان دهنده این است که کدبندی از یک مقدار صحیح منفی می‌باشد.

مثال:

Integer -27066
Step 1: 1101001 10111010
Step 2: 01101001 10111010
Step 3: 10010110 01000101
Step 4: 10010110 01000110

برای کدبندی، بیت مهم از مقدار عدد صحیح کدبندی شده مشخص می‌کند که آیا این مقدار مثبت است یا منفی.

1 - Twos-complement

اگر یک مقدار مثبت باشد، تبدیل روی بیت‌های باقیمانده که کم‌ارزش‌ترین بیت در موقعیت ۰ قراردارد، روی می‌دهد. مقدار صحیح دهدهی، مجموع مقادیر 2^n است که n، عدد موقعیت است، یا:

$$\sum_{n=0}^{p-1} 2^n$$

اگر یک مقدار منفی باشد، تبدیل بر روی بیت‌های باقیمانده که کم‌ارزش‌ترین بیت در موقعیت ۰ قراردارد، روی می‌دهد. اولین مرحله، ایجاد یک عدد صحیح دهدهی به عنوان مجموع مقادیر 2^n می‌باشد. دومین مرحله، این را به عنوان یک عدد صحیح دهدهی مثبت در نظر می‌گیرد که از آن، مقدار 2^p کم می‌شود که P عدد موقعیت بیت مهم است که مشخص می‌کند این یک عدد صحیح منفی است. به عنوان یک معادله، اینطور است:

$$\sum_{n=0}^{p-1} 2^n - 2^p$$

مثال:

```
10010110        01000110
1                               indicates -ve
 0010110        01000110    =   5702
2^P = 2^15 =                    32768
5702 - 32768 = -27066
```

الف-۲-۷ کدبندی مقدار OBJECT IDENTIFIER

کدبندی OBJECT IDENTIFIER باید اولین باشد تا مطابق با استاندارد ISO/IEC 8825-1 باشد. مقدار Object Identifier به صورت یک دنباله مقادیر ردیفی هشتایی به شکل زیر کدبندی می‌شود:

۱. دو قوس اول درخت ثبت، به صورت یک عدد صحیح با استفاده از این فرمول کدبندی می‌شوند:

$40f + s$

که f= مقدار اولین قوس

s = مقدار دومین قوس

۲. مقدار "n" از هر قوس اضافی، در یک فیلد- بیت- ردیفی- هشتایی[1] کدبندی می‌شود. این کار مطابق زیر برای مقادیر "n" انجام می‌گیرد:

الف- برای n<128

مقدار دهدهی به دودویی تبدیل می‌شود و در یک هشتایی کدبندی می‌شود. بنابراین بیت ۸ به عدد صفر تنظیم می‌شود.

ب- برای $128 \le n < 16384$

مقدار دهدهی به یک دودویی تبدیل می‌شود و به دو رشته ۷- بیتی تقسیم‌بندی فرعی می‌شود: بیت ۷ تا بیت ۱، بیت ۱۴ تا بیت ۸. هر یک از این رشته‌های بیت جدید در یک هشتایی

1 - Octet-aligned-bit-field

کدبندی می‌شود، که بیت ۸ از اولین هشتایی، به ۱ تنظیم می‌شود و بیت ۸ از آخرین هشتایی، به ۰ تنظیم می‌شود.

ج- برای $n \geq 16384$

مقدار دهدهی به دودویی تبدیل می‌شود و به رشته‌های ۷- بیتی تقسیم‌بندی فرعی می‌شود: بیت ۷ تا بیت ۱، بیت ۱۴ تا بیت ۸، بیت ۲۱ تا بیت ۱۵، و همین‌طور ادامه دارد. هر یک از این رشته‌های بیت جدید در یک هشتایی کدبندی می‌شود که بیت ۸ از اولین هشتایی، به ۱ تنظیم می‌شود، بیت ۸ از آخرین هشتایی، به ۰ تنظیم می‌شود و بیت ۸ از هشتایی‌های رابط، به ۱ تنظیم می‌شود. مثال زیر این فرآیند جریان را نشان می‌دهد:

مثال:

1. Value = 91234_{10}
   = $1\ 01100100\ 01100010_2$
2. Split into 7-bit strings
   0000101 1001000 1100010
3. Add prefix bits 0 for last octet
   1 for preceding octet(s)
   10000101 **1** 001000 **0**1100010

با استفاده از این روش، طول هر قوس مولفه OBJECT IDENTIFIER، خود اعلان است. اولین هشتایی همیشه دو قوس اول را مشخص می‌کند. هر قوس بعدی توسط یک هشتایی مشخص می‌شود اگر بیت مهم هشتایی بعدی ۰ باشد؛ و توسط چند هشتایی مشخص می‌شود اگر بیت مهم ۱ باشد، گروهی از هشتایی‌ها با هشتایی پایان می‌یابد که بیت اصلی آن معادل صفر است. مقدار قوس در ترتیب مقادیر ۷بیتی کدبندی می‌شود.

مثال:

[00101000]	1[1111000] 0[1001010]	0[0000001]
(1 x 40) + 0	15434	1
1 0	15434	1

اگرچه تعداد قوس ها، یک OJBECT IDENTIFIER از هر طول را قبول می‌کنند، ولی استاندارد ISO/IEC 15961: 2004 طول مقدار کدبندی شده را به حداکثر ۱۲۷ هشتایی محدود می‌کند. این محدودیت گذاشته شده تا الزامات کدبندی بر روی برچسب RF و ساختار Logical Memory را رعایت کند.

یادآوری- این شرط بر روی طول کدبندی شده OJBECT IDENTIFIER می‌باشد و نه تعداد قوس‌ها. همچنین باید توجه کردکه یک OJBECT IDENTIFIER کدبندی شده در هشتایی ۱۲۷ بسیار غیرمتحمل است.

الف-۲-۸ کدبندی یک مقدار OCTET STRING

اگرچه Basic Encoding Rules از استاندارد ISO/IEC 8825-1 امکان هر دو کدگذاری را می‌دهد ولی استاندارد ISO/IEC 15961: 2004 تنها کدبندی اولیه یک مقداری OCTET STRING را پشتیبانی می‌کند.
کدبندی اولیه شامل صفر، یک یا چند هشتایی می‌باشد که معادل در مقدار با هشتایی‌ها در مقدار داده کاربردی است. هشتایی‌های کد شده هم‌ردیف با مقدار داده‌ها ظاهر می‌شوند و بیشترین مقدار بیت هشتایی در هر دو نمونه کد شده و داده، ردیف می‌شود.

یادآوری- این بدین معناست که، برای سامانه‌های باز، توالی هشتایی و ترتیب بیت باید خروجی باشد و سامانه به‌طوردقیق همان ورودی را با سامانه ارسال دریافت کند. این وظیفه استانداردهای کاربردی است که الزامات برای توالی را مشخص کنند.

الف-۲-۹ کدبندی یک مقدار SEQURNCE

کدبندی یک مقدار SEQURNCE باید مطابق با استاندارد ISO/IEC 8825-1 ساخته شود. هشتایی‌های مضمون‌ها باید شامل کدبندی کامل TLV از یک مقدار داده باشد که این داده می‌تواند هر کدام از Types فهرست شده در تعریف ASN.1 از یک SEQURNCE Type ویژه باشد. مقادیر باید به ترتیب ظاهرشدنشان در تعریف باشند. اگرچه استاندارد ISO/IEC 8825-1 قواعد اختیاری برای Types را با کلید واژه‌های "OPTIONAL" یا "DEFAULT" در تعریف ANS.1 منظور می‌کند، استاندارد ISO/IEC 15961: 2004 به تمام Types در SEQURNCE نیاز دارد تا در کدبندی ساخته شده ظاهر شوند.

مثال:

ASN.1 definition
SEQUENCE {orderNumber OCTET STRING, product OCTET STRING, quantity INTEGER}

with the values:
{orderNumber "ABC1234", product "widget", quantity "12"}

is encoded as:

T = SEQUENCE	L			
30_{16}	14_{16}			
		T = OCTET STRING	L	V
		04_{16}	07_{16}	
			"ABC1234"	
		T = OCTET STRING	L	V
		04_{16}	06_{16}	
			"widget"	
		T = INTEGER	L	V
		02_{16}	01_{16}	$0C_{16}$

الف-۲-۱۰ کدبندی یک مقدار SEQURNCE OF

نوع SEQURNCE OF همان برچسب ANS.1 (UNIVERSAL 16) به‌عنوان نوع SEQUENCE است. بنابراین، قواعد کدبندی مشابه مورد پذیرش است. کدبندی مقادیر SEQURNCE OF باید مطابق با استاندارد

ISO/IEC 8825-1 ساخته شود. هشتایی‌های مضمون‌ها باید شامل کدبندی کامل TLV از هر مقدار باشد، از جمله کدبندی مکرر برچسب UNIVERSAL ار اقلام کدگذاری شده.

مثال:

ASN.1 definition
SEQUENCE {productCode OCTET STRING, }

with the three values:
{productCode "ABC1234", "X6789Y", "PQR12345"}

is encoded as:

T = SEQUENCE L
30_{16} $1B_{16}$

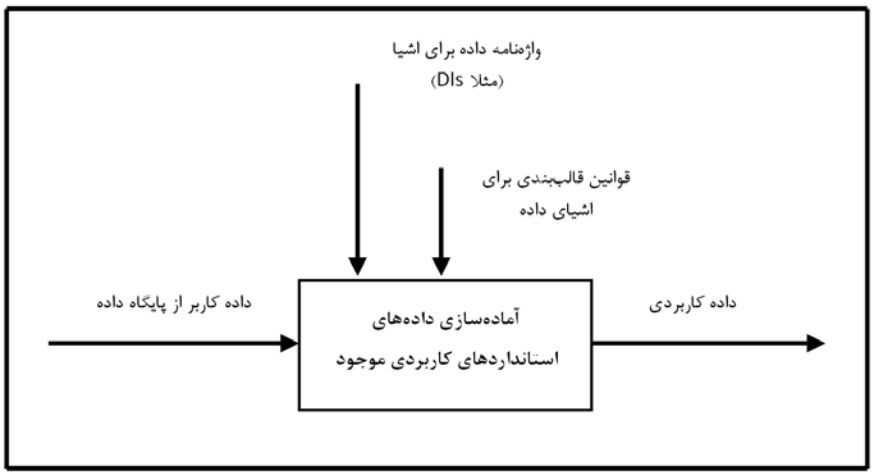
T = OCTET STRING L V
04_{16} 07_{16}
"ABC1234"
T = OCTET STRING L V
04_{16} 06_{16}
"X6789Y"
T = INTEGER L
V
04_{16} 08_{16} "PQR12345"

پیوست ب
(اطلاعاتی)
تطبیق دادن قالب‌های داده تعریف شده

این استاندارد بر این مبنا تهیه شده است که پروتکل مبتنی بر شی متفاوت از پروتکل‌های پیغام و قواعد نحوی چند استاندارد کاربردی AIDC می‌باشد. بنابراین اشیا داده اصلی باید مرتبط با استاندارد کاربردی ارائه شوند، مثلا برحسب:

- **Object** داده‌ها در فرهنگ لغات داده‌ها پشتیبانی شود.
- قالب داده‌ها (مثلا عددی، الفبایی)، از جمله طول آن.
- ترکیب‌های Objects داده‌ها اعم از اینکه معتبر یا غیرقانونی هستند.

این ویژگی‌ها خارج از حوزه کاربرد این استاندارد می‌باشند و بر عهده برنامه کاربردی هستند.
چند فرآیند تبدیل لازم است تا اینکه سامانه‌های کاربردی بتوانند داده‌ها و شناسه‌ها را در قالب مشخص شده در این استاندارد مدیریت کنند. می‌توان دو مسیر پیاده‌سازی مستقل داشت. یک مسیر برای نوشتن داده‌ها و مسیر دیگر برای خواندن داده‌ها.
برای برخی از کاربردها مهم، بعضی قواعد سخت[1] در این مورد وجود دارند که چه چیزی داده‌های قانونی را تشکیل می‌دهد. زمانی که از قواعد نحوی پیغام موجود استفاده می‌شود، نرم‌افزاری وجود دارد که اطمینان حاصل کند که با این قالب تطابق دارد. کاربران باید مطمئن شوند که زمانی که یک روش مبتنی بر Object را برای نوشتن داده‌ها پیاده می‌کنند، این داده‌ها قواعد اصلی را دنبال می‌کنند. شکل ب- ۱- مدل جریان داده: آماده‌سازی داده‌های استانداردهای کاربردی موجود - این طرح کلی را شرح می‌دهد.



شکل ب- ۱- مدل جریان داده: آماده‌سازی داده‌های استانداردهای کاربردی موجود

1 - Rigorous

زمانی که داده‌ها از یک برچسب RFID خوانده می‌شوند، یک فرآیند مشابه لازم است.
فناوری‌های AIDC که بر مبنای فناوری یک بار نوشتن و چند بار خواندن (WORM)[1] است، به این واقعیت مربوط می‌شود داده هنگامی‌که نوشته می‌شوند، همان چیزیست که خوانده می‌شود. یعنی قواعد نحوی پیغام در حامل داده‌ها کدبندی می‌شود. امکانات خواندن- نوشتن RFID و طبیعت مبتنی بر Object پروتکل داده‌ه از این استاندارد، بدین معنا می‌باشد که یک قواعد نحوی تعریف شده باید براساس ساختار Object-Identifier ساخته شود. در حالی‌که یک الزام برای داده‌های خروجی با یک قواعد نحوی داده‌های ویژه وجود دارد (مثال همانند استاندارد ISO/IEC 15434) سپس یک پودمان تبدیل لازم است تا مجموعه‌ای از Object-Identifiers و data Objects را در قالب پیغام به درستی نگاشت کنند که برای کاربرد این استاندارد لازم می‌باشد. این باید تبدیل قواعد تبدیل برای چندین داده کاربردی را الزام کند.
همچنین، قواعد نحوی پیغام از استانداردهای کاربردی تعریف شده باید ساخته شوند. بطور کلی این فرآیند، به تمام آن‌ها نیاز دارد اما قوس نهایی Object-Identifier دور انداخته می‌شود و جدا کننده‌های داده‌ها (موافق با استاندارد کاربردی) باید به درستی درج شوند. برای قواعد دقیق به استانداردهای کاربردی مناسب مراجعه کنید.
مرحله توسعه منطقی، برای استاندارد کاربردی می‌باشد تا دستورالعمل‌هایی را تهیه کند که خروجی را براساس قواعد نحوی انتقال بپذیرد.

1 - Write-once-read-many-times

پیوست پ
(اطلاعاتی)
اشیا داده مرتبط

قواعد نحوی پیغام می‌توانند از روش‌های بازگشتی یا ایجاد حلقه استفاده کنند تا توالی تکراری از داده‌های مرتبط را بسازند (مثال کمیت فردی و تعداد دسته‌ای مربوط به کدهای محصول مختلف). زمانی که پیغام کامل تجزیه می‌شود[1]، این قانون نحوی نقاط مرزی را مشخص می‌کند بنابراین ویژگی‌ها به درستی به کد اولیه ارتباط داده می‌شوند.

با یک سامانه مبتنی بر Object (از قبیل Data protocol از این استاندارد و استاندارد ISO/IEC 15962) که در یک سطح پایه عمل می‌کند، یک خطر ایجاد پیوندهای اشتباه وجود دارد (یعنی کد محصول الف ممکن است به کمیت محصول ب متصل شود). این مسئله را می‌توان با استفاده از یکی از روش‌هایی که در زیر شرح داده می‌شود حل کرد. این روش‌ها باید تنها زمانی انتخاب شوند که با استاندارد کاربردی مربوط به موضوعی که مدیریت می‌شوند مرتبط باشند. این توضیحات تعداد اقلام داده‌های ساخته شده را در هر برچسب RFID تا ۲۵۵ محدود می‌کند، اما قواعد متفاوت ممکن است ایجاد شود اگر تعداد بیشتری از ساختار لازم باشد. هر قانون برای Data protocol کامل این استاندارد و استاندارد ISO/IEC 15962 شفاف و روشن است و بنابراین پردازش باید به عنوان بخشی از این برنامه کاربردی پیاده شود. این گزینه‌ها در این استاندارد قرار می‌گیرند تا روش‌های قوی و مؤثری را شرح دهند که با استفاده از ساختار درختی Object-Identifier از یک فرآیند اکتساب داده‌های مبتنی بر Object محافظت می‌کنند.

پ-۱ روش‌های الحاق

Object-Identifier ویژه را می‌توان ساخت که یک مجموعه تعریف شده‌ای از ویژگی‌ها را بطور الحاقی متصل کنند.

مثال:

کوچکترین قوس ۲۵۴ =

- شماره ترتیب ۱ هشتایی
- کد محصول ۸ هشتایی
- کمیت ۱ هشتایی
- عدد دسته ۴ هشتایی

در این مورد، اولین بایت **Object**، شماره ترتیب، یک داده Object مشابه را از دیگری تشخیص می‌دهد.

1 - Pars

با استفاده از این روش، هر ترتیب مختلفی از اقلام اصلی که ساختار الحاقی را ایجاد می‌کنند یک گره نهایی متفاوتی دارند. بنابراین الحاق محصول+ کمیت+ تاریخ انقضا کوچکترین مقدار کمان خود را دارد که متفاوت از محصول+ کمیت+ دسته می‌باشد.

این روش مناسب‌تر است زمانی که ترکیبات ثابتی از اقلام باید ساخته شود و طول هر **Object** ثابت است.

پ- ۲ روش توسعه شناسه Object

Object-Identifier اصلی را می‌توان با افزودن یک قوس نهایی جدید توسعه داد و این به عنوان یک مقدار «پیوندی» می‌باشد.

مثال:

سه اقلام زیر باید پیوند داده شوند:

- کد محصول- قوس نهایی ۴۸
- کمیت- قوس نهایی ۱۷
- دسته- قوس نهایی ۲۰

فرض کنید که دو محصول مختلف وجود دارند که جزئیات آن‌ها بر روی برچسب RFID کدبندی می‌شوند. بنابراین توسعه‌های پیوندی ۱ و ۲ به کاربرده می‌شوند. شش **Object-Identifier** شخصی کدبندی می‌شوند:

... 48 1
.... 48 2
.... 17 1
.... 17 2
.... 20 1
.... 20 2

مقدار توسعه برای پیوند Objects مختلف استفاده می‌شود که به عنوان یک ترکیب منطقی می‌باشد.

این روش بیشتر زمانی مناسب است که ترکیبات مختلف زیادی از اقلام باید ساخته شود و طول یک Object، حداقل، ممکن است بین رخدادها متفاوت باشد.

روش توسعه برای Objects مرتبط و موجودیت‌های فیزیکی مربوطه آن‌ها شبیه به طرح ب برای استفاده از امنیت داده‌ها می‌باشد (به پیوست پ-۱ مراجعه کنید). بنابراین برای هر یک از **Object-Identifier**، این روش باید تنها برای امنیت داده‌ها و یا پیوند موجودیت‌های فیزیکی به کاربرده شود.

پیوست ت
(اطلاعاتی)
مسائل امنیت داده‌ها

اگرچه امنیت داده‌ها فراتر از حوزه کاربرد این استاندارد و استاندارد ISO/IEC 15962 می‌باشد، توصیه زیر تهیه می‌شود تا نشان دهد که چگونه ویژگی‌های Data protocol ممکن است استفاده شوند تا داده‌های مطمئن‌تری به دست آید.

ت-۱- مسائل Object-Identifier

داده‌های رمز شده باید به **Object-Identifier** انحصاری خودشان مربوط شوند. این اطمینان می‌دهد که کاربران مجاز می‌توانند داده‌های رمز شده را تشخیص دهند، اما این حقیقت را به کاربران دیگر اعلان نمی‌کنند. خود **Object** به آسانی همراه با مجموعه **Compact-Parameter** به عنوان **Application-Defined** ظاهر می‌شود (به زیربند ۷-۳-۵ مراجعه کنید).

یک روش (با نام طرح الف برای مراجعه بعدی) تولید **Object-Identifier**، به این خاطر است که یک قوس نهایی را در همان سطح از تمام قوس‌های نهایی دیگر در سامانه کاربردی داشته باشیم. این یک پذیرش سطح سامانه از امنیت داده‌ها می‌باشد و نیاز است که تمام کاربران مجاز بدانند که داده رمزنگاری شده است؛ هرچند که این قواعد نباید بطور عمومی اعلان شوند.

روش دیگر (با نام طرح ب برای مراجعه بعدی) برای تولید یک **Object-Identifier** انحصاری، شناسایی داده‌های رمزنگاری شده می‌باشد تا **Object-Identifier** داده‌های ساده (رمزدار نشده) را توسعه دهد و یک قوس پایین‌تر دیگری را اضافه کند. این روش می‌تواند از دو طرف بین فرستنده و کاربر(ان) مجاز یا در سطوح سامانه برای تمام کاربران مجاز مورد پذیرش واقع شود. این روش برای تعریف نوع رمزنگاری، کلیدهای انتخابی و مانند اینها استفاده می‌شود.

مثال:

0 1 15961 nn nn	شی ساده
0 1 15961 nn nn 1	شی رمزنگاری شده
0 1 15961 nn nn 2	نوع رمزنگاری
0 1 15961 nn nn 3	کلید

طرح ب شبیه به طرح پیشنهادی برای Objects مربوطه و موجودیت‌های فیزیکی مربوط به آن‌ها می‌باشد (به پیوست پ-۲ مراجعه کنید). بنابراین برای هر یک از **Object-Identifier**، این روش باید فقط در امنیت داده‌ها یا در پیوند موجودیت‌های فیزیکی به کار برده شود.

ت-۲ Object داده

Object که شامل داده‌های کاربردی می‌باشد باید پس از رمزدار کردن، **Compact-Parameter** خود را به Application-Defined تنظیم کند.

Object اصلی باید توسعه پیدا کند تا شامل یک فیلد داده‌های از پیش تعریف شده و یا امضای قسمت نوشتن مجاز باشد. این کمک می‌کند تا چنانکه هر اصلاح غیرمجاز **Object** رمزدار شده بدون دسترسی به کلید خصوصی انجام شود، یکپارچگی داده‌ها شما را مطمئن کند که با بیشترین احتمال، این امضای مجاز را خراب خواهد کرد. این کمک می‌کند که شما تشخیص دهید که این **Object** بدون اجازه تغییر کرده است.

اگر یک **Object-Identifier** متفاوت به داده‌های رمزدار شده تخصیص داده شود (طرح الف بالا)، باید توسعه و تعمیم پیدا کند تا شامل هشتایی‌های رمزدار نشده اضافی شود که طرح رمزنگاری و/یا انتخاب از یک مجموعه کلیدها را تعریف می‌کند.

ت-۳ استفاده از Tag ID

می‌توان از روش Tag ID یکتا به عنوان یک مؤلفه در یک سامانه امنیتی استفاده کرد (همان‌طور که برای نوع‌های گوناگون در استاندارد ISO/IEC 18000 تعریف شده است). tag ID در برچسب RFID منحر به فرد است تا از سایرین تشخیص داده شود. به‌طورمعمول در هر مرحله اولیه‌ای از ساخت، برچسب RFID با استفاده از روش‌های قوی‌تری ساخته می‌شود تا بتوان برای نوشتن داده‌ها در Logical Memory Map استفاده کرد. همین‌طور برای افزایش اعتبار داده‌ها می‌توان استفاده کرد. به دلیل اشتباهی گرفتن احتمالی با **Singulation-Id** پروتکل داده‌ها (که می‌تواند از Tag-ID استفاده کند)، در قسمت‌های دیگر این پیوست، Tag-ID درون مدار مجتمع، با عنوان Tag-ID 18000 اشاره می‌شود.

همچنین اگر Tag-ID 18000 همانند **Singulation-Id** بخشی از اطلاعات سامانه عمل کند، Tag-ID 18000 در هر مرحله اولیه‌ای از ارتباطات، با برچسب RFID تهیه می‌شود. اگر Tag-ID 18000 به عنوان بخشی از اطلاعات سامانه‌ها تهیه نشود، فرمان‌های دیگری باید آن را بخوانند.

یک روش، الحاق کردن مقدار Tag-ID 18000 به داده‌های اصلی **Object** و رمزدار کردن تمام **Object** توسعه یافته می‌باشد. زمانی که توسط یک کاربر مجاز کشف رمز می‌شود، Tag-ID 18000 در داخل **Object** توسعه یافته را می‌توان با Tag-ID 18000 واقعی مقایسه کرد تا بررسی کنیم که آن‌ها یکسان هستند. این ممکن است یا در طرح الف یا طرح ب برای ساخت **Object-Identifier** به کار برده شود، که در بالا توصیف شد.

روش دیگر، استفاده از Tag-ID 18000 می‌باشد تا کلید اصلی را برای رمزدار کردن و کشف رمز اصلاح کنیم. این کار را می‌توان برای کلیدهای دودویی انجام داد از قبیل DES، که در آنجا Tag-ID 18000 ممکن است با کلید اصلی یای انحصاری (*exclusive or-ed*) شود.

قبل از استفاده از این روش، پیاده‌سازها باید بررسی کنند که این نوع اصلاح کلید، روش رمزدارکردن را از بین نبرد.

ت-۴ توصیه در مورد روش‌های کلید عمومی رمزدار کردن

الگوریتم‌های کلید عمومی به کلیدهای طولانی‌تری نیاز دارند تا استحکام و مقاومتی را همانند کلیدهای متقارن فراهم کنند. برای مثال یک رمز که با کلید عمومی ۵۱۲ بیتی رمزنگاری شده مقاومت و استحکامی معادل یک رمز کلید متقارن ۶۴ بیتی دارد. این طول رمز ممکن است مانع از استفاده رمزهای کلید عمومی در برچسب‌های ظرفیت کمتر شود.

اگر برای رمزنگاری از روش کلید عمومی استفاده شود، در صورتی‌که کلید خصوصی برای رمزنگاری داده‌ها و کلید عمومی برای کشف رمز داده‌ها استفاده شود، امنیت داده‌ها ممکن است به خطر بیافتد. همچنین یکپارچگی داده‌ها به خطر می‌افتد اگر کلید عمومی برای رمزدار کردن داده‌ها و کلید خصوصی برای کشف رمز داده‌ها استفاده شود. رمزدار کردن دوتایی[1] یا ابزار[2] دیگری باید استفاده شود تا از امنیت و یکپارچگی داده‌ها مطمئن شویم.

کلید عمومی نباید در برچسب، کدبندی شود مگر اینکه به عنوان یک بخش غیرمجاز قفل شود که ممکن است با نوشتن مجدد[3] کلید عمومی بوسیله کلید دیگر و استفاده از کلید خصوصی مشابه دیگر برای رمزنگاری داده‌های تغییر یافته در Object، از یکپارچگی داده تخلف کند.

1 - Double Encryption
2 - Means
3 - Overwrite

پیوست ث

(اطلاعاتی)

دستورات و پاسخ‌های اصلی با استفاده از قواعد نحوی انتزاعی ASN.1

زیربندهای زیر۱۶ پودمان اصلی را با استفاده از قواعد نحوی انتزاعی ASN.1 نشان می‌دهند. مرجع‌های متقابل[1] گاهی اوقات با یک تغییر نام، برای بندهای الزامی تهیه می‌شوند که اکنون جایگزین آن‌هایی می‌شوند که در پودمان‌های اصلی بوده‌اند.

ث-۱ ConfigureAfiModules

ConfigureAfiModules شامل یک commandModule و responseModule مربوط می‌باشد که به تحقیق‌کننده آموزش می‌دهد تا **ApplicationFamilyId** (شامل خانوداه فرعی[2]) را در برچسب RFID بنویسد. یک الزام اساسی این دستور این است که تنها یک برچسب RFID باید به ازای هر فرمان برنامه‌نویسی شود تا مطمئن شویم که این فرآیند پیکربندی قوی و مؤثر می‌باشد، به خصوص در محیط‌هایی که بیشتر از یک نوع برچسب RFID ممکن است وجود داشته باشد.

قواعد نحوی انتزاعی برای ConfigureAfiModules در جدول ث-۱- ConfigureAfiModules- ارائه می‌شوند.

جدول ث-۱- ConfigureAfiModules

```
-- Configure AFI
-- The ConfigureAfiCommand instructs the interrogator to write the AFI (Application
-- Family Identifier, including the sub-family) into the RFID tag. The interrogator shall
-- lock the AFI if the Lock flag is set to true.
ConfigureAfiCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) configureAfi(1)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN
ConfigureAfiCommand ::= SEQUENCE {
tagId OCTET STRING(SIZE(0..255)),
-- See Clause 7.2.1 (now renamed Singulation-Id) for detailed
-- specification. TagId shall be provided by the Tag Driver for the
    -- purposes of identifying the RFID tag unambiguously
    -- for at least the period of a transaction.
applicationFamilyId ApplicationFamilyId,
afiLock BOOLEAN
-- If set to TRUE, the interrogator shall lock the AFI
}
```

1 - Cross-references
2 - Sub-family

جدول ث-۱- ادامه

```
ApplicationFamilyId ::= SEQUENCE {
        applicationFamily INTEGER {
                                all(0), -- address all families
                                -- values 1 - 8 reserved for definition by SC17
                                afiBlock9(9),
                                afiBlockA(10),
                                afiBlockB(11),
                                afiBlockC(12)
                                -- values 9 to 12 defined as per Annex B of this
                                        -- International standard
                                -- values 13 to 15 reserved for definition by ISO/IEC
                                } (0..15),
        applicationSubFamily INTEGER {
                                all(0),-- This value shall not be encoded in the RFID tag, and
                                -- shall only be used in a command to signal that the
                                   -- interrogator shall address all subfamilies within the
                                   -- selected family.
                                   -- NOTE: This has little utility for this Data Protocol, but
                                   -- is retained for compatibility with SC17 smart card
        commands
                                   asf1-annex (1), -- values 1 to 15, for applicationFamily 9
                                   -- to 12,defined as per Annex B of this part of ISO/IEC 15961
                                   asf2-annex (2),
                                   asf3-annex (3),
                                   asf4-annex (4),
                                   asf5-annex (5),
                                   asf6-annex (6),
                                   asf7-annex (7),
                                   asf8-annex (8),
                                   asf9-annex (9),
                                   asfA-annex (10),
                                   asfB-annex (11),
                                   asfC-annex (12),
                                   asfD-annex (13),
                                   asfE-annex (14),
                                   asfF-annex (15)
                                   } (0..15)
-- ApplicationFamilyId is stored as a single OCTET within system information on the tag.
-- ApplicationFamilyId allows tags to be grouped according to specific families and
-- allows any such family of tags to be selectively addressed by the application. RFID tag
-- vendors may implement mechanisms in the Tag Driver and air interface specifically for
                                -- selective addressing of RFID tags by ApplicationFamilyId.

}
END
ConfigureAfiResponse
        {iso(1) standard(0) rfid-data-protocol(15961)responseModules(127) configureAfi(1)}
DEFINITIONS
```

جدول ث-۱- ادامه

```
EXPLICIT TAGS ::=
BEGIN

ConfigureAfiResponse ::= SEQUENCE {
        completionCode INTEGER {
                        noError(0),
                        afiNotConfigured(1),
                        afiNotConfiguredLocked(2),
                        afiConfiguredLockFailed(3),
                        tagIdNotFound(8),
                        executionError(255)
                        },
        executionCode INTEGER
-- See Clause 9.3 and notes in this syntax for a full list of
    -- executionCodes
}
END
```

elementNamesهای زیرکه در این پودمان‌ها استفاده شده‌اند، در جای دیگر از این استاندارد تعریف می‌شوند، که به تفصیل شرح داده شده است:

AfiLock (به ۷-۴-۳ مراجعه کنید)

applicationFamily (به زیربند ۷-۲-۲ مراجعه کنید)

applicationFamilyId (به زیربند ۷-۲-۲ مراجعه کنید)

applicationSubfamily (به زیربند ۹-۲ مراجعه کنید)

executioncode (به زیربند ۹-۳ مراجعه کنید)

tagId (به زیربند ۷-۲-۱ مراجعه کنید)

ث-۲ ConfigureStorageFormatModule

ConfigureStorageFormatModule شامل یک commandModnle و responseModule مربوطه می‌باشد که به تحقیق‌کننده آموزش می‌دهد تا **storageFormat** (**accessMethod** و **dataFormat**) را در برچسب RFID بنویسد. همچنین این دستور به تحقیق‌کننده آموزش می‌دهد که به برچسب RFID در Logical Memory Map مقدار اولیه بدهد تا هر داده‌ای را که از قبل در آنجا ذخیره شده بود را پاک کند. یک الزام اساسی این فرمان این است که تنها یک برچسب RFID باید به ازای هر فرمان برنامه‌نویسی شود تا مطمئن شویم که فرآیند شکل‌بندی قوی و مؤثر است، به خصوص در محیط‌هایی که بیشتر از یک نوع برچسب RFID می‌تواند وجود داشته باشد.

اگر **accessMethod** (پیوست شده در **StorageFormat**) به عنوان فهرست راهنما مشخص شود، تحقیق‌کننده باید ساختار اولیه فهرست راهنما را ایجاد کند.

قواعد نحوی انتزاعی ASN.1 برای ConfigureStorageFormatModules در جدول ث-۲- ConfigureStorageFormatModules- ارائه می‌شود.

جدول ث-۲- ConfigureStorageFormatModules

```
-- Configure StorageFormat
-- The ConfigureStorageFormatCommand instructs the interrogator to write the
-- StorageFormat into the RFID tag, and to initialise the tag logical memory
-- map. The interrogator shall erase all the application memory, and if the
-- directory format is specified by the StorageFormat, it shall create the
-- initial directory structure. The interrogator shall lock the
-- StorageFormat if the Lock flag is set to true
ConfigureStorageFormatCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126)
configureStorageFormat(2)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

ConfigureStorageFormatCommand ::= SEQUENCE {
        tagId                OCTET STRING(SIZE(0..255)),
                             -- See Clause 7.2.1 (now renamed Singulation-Id) for detailed
                             -- specification. TagId shall be provided by the Tag Driver for the
                             -- purposes of identifying the RFID tag unambiguously
                             -- for at least the period of a transaction.
        storageFormat            StorageFormat,
        storageFormatLock        BOOLEAN
                                 -- If set to TRUE, the interrogator shall lock the
                                 -- StorageFormat
}
StorageFormat ::= SEQUENCE {
        accessMethod    INTEGER {
noDirectory(0),
                             directory(1),
                             selfMappingTag(2) -- Access to objects is via high
                             -- level commands to the RFID tag and the internal
                             -- structure of the memory inside the RFID tag is not
                             -- defined
                             } (0..3),
        dataFormat       INTEGER {
                             notFormatted(0), -- Not formatted according
                             -- to this part of ISO/IEC 15961
                             fullFeatured(1), -- Supports any type of
                             -- data based on full OID
                             rootOidEncoded(2), -- Supports any type of
                             -- data with a common root-OID
                             iso15434(3), -- root-OID is defined as
```

جدول ث-۲- ادامه

```
                    -- {1 0 15434}
                    iso6523(4), -- Supports data belonging to one or
                        -- more International Code Designators compliant
                        -- with ISO/IEC 6523-1, root-OID is defined as
                    -- (1 0 6523)
                    iso15459(5), -- Supports unique item identifiers
                        -- compliant with ISO/IEC 15459, root-OID is
                        -- defined as (1 0 15459)
                    iso15961Combined(8), -- Supports combinations of
                        -- formats of ISO/IEC 15961, root-OID is defined
                        -- as {1 0 15961}
                    ean-ucc(9), -- Supports data of the EAN-UCC system,
                        -- root-OID is defined as {1 0 15961 9}
                    di(10), -- Supports Data Identifiers (as referred to
                        -- in ISO/IEC 15418), root-OID is implied to be
                        -- {1 0 15961 10}
                    iata(12) -- Supports IATA baggage handling data elements,
                    -- root-OID is defined as {1 0 15961 11}
                    } (0..31)
}
END
ConfigureStorageFormatResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) configureStorageFormat(2)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

ConfigureStorageFormatResponse ::= SEQUENCE {
        completionCode INTEGER {
                                noError(0),
                                storageFormatNotConfigured(4),
                                storageFormatNotConfiguredLocked(5),
                                storageFormatConfiguredLockFailed(6),
                                tagIdNotFound(8),
                                executionError(255)
                                },
        executionCode INTEGER
                                -- See Clause 9.3 and notes in this syntax
                                -- for a full list of executionCodes
}
END
```

elementNamesهای زیرکه در این پودمان‌ها استفاده شده‌اند، در جای دیگر از این استاندارد تعریف می‌شوند، که به تفصیل شرح داده شده است:

accessMethod (به زیربند ۷-۲-۴ مراجعه کنید)

completionCode (به زیربند ۹-۲ مراجعه کنید)
dataFormat (به زیربند ۷-۲-۵ مراجعه کنید)
executionCode (به زیربند ۹-۳ مراجعه کنید)
storageFormat (به زیربند ۷-۳-۲ مراجعه کنید)
storageFormatLock (به ۰ مراجعه کنید)
tagId (به زیربند ۷-۲-۱ مراجعه کنید)

ث-۳ InventoryTagsModules

InventoryTagsModules شامل یک completionModule و responseModule مربوطه می‌باشد که به تحقیق‌کننده آموزش می‌دهد که یک مجموعه خاصی از برچسب‌های RFID موجود در فیلد عملیاتی آن مشخص کند.

قواعد نحوی انتزاعی ASN.1 برای InventoryTagsModules در جدول ث-۳- InventoryTagsModules- ارائه می‌شود.

این دستور به مقدار **applicationFamilyId** نیاز دارد تا برچسب‌های RFID متعلق به یک کلاس خاص را انتخاب کند، که بطور نمونه شامل داده‌هایی است که به یک دامنه تعریف شده تعلق دارد و/یا شامل یک **ObjectId** تعریف شده می‌باشد.

معیار انتخاب ثانویه دیگر (**identifyMethod**) تعیین می‌کند که چه تعداد برچسب‌های RFID، مطابق با معیار انتخابی مشخص شده **applicationFamilyId** می‌باشند که باید قبل از تهیه پاسخ شناسایی شوند. یک سازوکاری که می‌توان استفاده کرد تا هر برچسب RFID که وارد ناحیه عملیاتی می‌شود را کشف کنیم، این است که متغیر InventoryAtLeast را به مقدار ۱ تنظیم کنیم. شرایط خاص را می‌توان تنها با تقبل کردن یک فهرست[1] جزئی اثبات کرد یعنی با استفاده از متغیرهای InventoryAtLeast یا InvertoryNoMereThan. یک تطبیق کمیت شناخته شده از تراکنش‌های قبلی (مثلا تشخیص اینکه تمام موضوعات یا داده‌ها در داخل یک محتوی[2] در واقع وجود دارند) با استفاده از متغیر InventoryExactly می‌تواند به دست آید. جزئیات گزینه‌ها در داخل پودمان در جدول ث-۲- InventoryTagsModules - ارائه می‌شوند.

این پاسخ شامل **numberOfTagsFound** و هویت‌های هر برچسب RFID توسط **tagId** آن می‌باشد.

1 - Inventory
2 - Container

InventoryTagsModules -۳-ث جدول

```
-- InventoryTags
-- The InventoryTagsCommand instructs the interrogator to inventory and to
-- identify all tags present in its operating area. Each tag is uniquely
-- identified by its TagId.


InventoryTagsCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) inventoryTags(3)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

InventoryTagsCommand ::= SEQUENCE {
        applicationFamilyId ApplicationFamilyId,
        identifyMethod          INTEGER {
                                        inventoryAllTags (0),
                                        inventoryAtLeast (1),
                                        inventoryNoMoreThan (2),
                                        inventoryExactly (3)
                                        }(0..15),
numberOfTags                    INTEGER (0..65535)
}
-- The ApplicationFamilyId separates fundamentally different types of application
-- data (see 7.2.2), and possibly particular objectIds. Specifying a
-- hex value xx (where x is a non-zero value) selects only the RFID tags that have
-- the required data content.
-- Specifying a hex value 00 selects all the RFID tags; this may be an appropriate
-- action to undertake a full inventory.
-- Specifying a hex value 0x, or x0, (where x is a non-zero value) might not be
-- logically sound because the RFID tags are from different applications and the x
-- value has different meaning.
-- If the identifyMethod is set to inventoryAllTags, the interrogator shall perform
-- a complete inventory of all tags present in its field of operation. The value of
-- numberOfTags is irrelevant and should be set to zero by the application.
-- If the identifyMethod is set to inventoryAtLeast, the interrogator shall perform
-- an inventory of the tags present in its field of operation and (possibly)
-- continue waiting until it has identified a number of tags equal to numberOfTags.
-- If the numberOfTags is set to 1, the Interrogator will wait until the first tag
-- has been detected. This is a mechanism to wait for a tag to enter the
-- interrogator field.
-- If the numberOfTags is set to more than 1, the Interrogator will wait
-- until the specified number of tags has been detected.
-- If the identifyMethod is set to inventoryNoMoreThan, the interrogator shall
-- initiate an inventory of the tags present in its field of operation and shall
-- return a response with a number of tags lower or equal to numberOfTags.
-- The interrogator may interrupt the inventory process when the numberOfTags has
```

```
-- been reached or may continue the inventory process till all tags have been read.
-- Note: This may be constrained by the air interface and anticollision
-- mechanism.
-- If the identifyMethod is set to inventoryExactly, the interrogator shall
-- initiate an inventory of the tags present in its field of operation and shall
-- return a response with the number of tags equal to numberOfTags. This command
-- parameter could be used to confirm the actual number of tagged items in a
-- container. The Interrogator will wait until the specified number of tags has
-- been detected. The interrogator may interrupt the inventory process when the
-- numberOfTags has been reached or may continue the inventory process till
-- all tags have been read.
-- Note: This may be constrained by the air interface and anticollision
-- mechanism.
-- Execution of this command with the arguments inventoryAtLeast and
-- inventoryExactly can cause the interrogator to wait until sufficient RFID tags
-- enter its field of operation; also the command response cannot be initiated
-- until after this delay. It is the responsibility of the application to
-- accommodate this potentiality.

ApplicationFamilyId ::= SEQUENCE {
        applicationFamily INTEGER(0..15),
        applicationSubFamily INTEGER(0..15)
        }
-- The code values are defined in the ConfigureAfiCommand.

TagId ::= OCTET STRING(SIZE(0..255))
        -- See Clause 7.2.1 for detailed specification
END
InventoryTagsResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) inventoryTags(3)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

InventoryTagsResponse ::= SEQUENCE {
        completionCode          INTEGER {
                                        noError(0),
                                        failedToReadMinimumNumberOfTags(23),
                                        -- for example, this could be due to a time-out
                                        failedToReadExactNumberOfTags(24),
                                        -- for example, this could be due to a time-out
                                        executionError(255)
                                        },
        executionCode           INTEGER,
                                        -- See Clause 9.3 and notes in this syntax
```

جدول ث-۳- ادامه

```
                                -- for a full list of executionCodes
        numberOfTagsFound INTEGER (1..65535),
        identities SEQUENCE OF TagId
        }

TagId ::= OCTET STRING(SIZE(0..255))
                -- See Clause 7.2.1 for detailed specification
END
```

elementNamesهای زیرکه در این پودمان‌ها استفاده شده‌اند، در جای دیگر از این استاندارد تعریف می‌شوند، که به تفصیل شرح داده شده است:

ApplicationFamily (به زیربند ۷-۲-۲ مراجعه کنید)

applicationFamilyId (به زیربند ۷-۲-۲ مراجعه کنید)

applicationSubfamily (به زیربند ۹-۲ مراجعه کنید)

completionCode (به زیربند ۹-۲ مراجعه کنید)

executionCode (به زیربند ۹-۳ مراجعه کنید)

identifyMethod (به زیربند ۷-۴-۲۳ مراجعه کنید)

identities (به زیربند ۷-۵-۲ مراجعه کنید)

numberOfTags (به زیربند ۷-۴-۴۳ مراجعه کنید)

numberOfTagsFound (به زیربند ۷-۵-۹ مراجعه کنید)

tagls (به زیربند ۷-۲-۱ مراجعه کنید)

ث-۴ AddSingleObjectModules

AddSingleObjectModules شامل commandModule و responseModule مربوطه می‌باشد که به تحقیق‌کننده آموزش می‌دهد که یک **Object**، **ObjectId** و پارامترهای مربوطه آن را در برچسب RFID در Logical Memory Map بنویسد. متغیرهای این فرمان را می‌توان استفاده کرد تا **ObjectId**، **Object** و پارامترهای مربوطه را قفل کنند و بررسی کنند که **ObjectId** بر روی برچسب RFID کدبندی نمی‌شود. تنها یک برچسب RFID باید به ازای هر فرمان برنامه‌نویسی شود تا مطمئن شویم که فرآیند نوشتن قوی و مؤثر است.

قواعد نحوی انتزاعی ASN.1 برای AddSingleObjectModules در جدول ث-۴- AddSingleObjectModules - ارائه می‌شود.

جدول ث-۴- AddSingleObjectModules

```
-- Add Single Object
-- The AddSingleObjectCommand instructs the interrogator to write an object, its
-- OID and associated parameters into the tag logical memory map.
-- NOTE: There is also an AddMultipleObjectsCommand.

-- If the checkDuplicate flag is set to TRUE, the interrogator shall verify, before
-- adding the object, that no object with the same OID already exists. If such
-- object exists, the interrogator shall not perform the Add Object function and
-- shall return the appropriate Completion Code.

-- If the Lock flag is set to TRUE, the interrogator shall lock the ObjectId, the
-- Object, its compaction scheme and associated parameters into the tag Logical
-- Memory Map


AddSingleObjectCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) addSingleObject(4)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

AddSingleObjectCommand ::= SEQUENCE {
        tagId               OCTET STRING(SIZE(0..255)),
                            -- See Clause 7.2.1 for detailed specification
        objectId            OBJECT IDENTIFIER,-- Full OID value
        avoidDuplicate BOOLEAN,
                            -- If set to TRUE, check for duplicate objectId
        object              OCTET STRING,
        compactParameter INTEGER {
                                    applicationDefined(0),
                                    -- The object shall not be processed through the -- data
                                    compaction rules of 15962 and remains unaltered
                                    compact(1),
                                    -- Compact object as efficiently as possible
                                    -- using 15962 compaction rules
                                    utf8Data(2)
                                    -- Data has been externally transformed from a 16-bit
                                    -- coded character set to a UTF-8 string. The object
                                    -- shall not be processed through the data compaction
                                    -- rules of 15962 and remains unaltered
                                    }(0..15),
        objectLock                  BOOLEAN
                                    -- If TRUE the interrogator shall lock the ObjectId, the
                                    -- Object, its compaction scheme and other features in the
                                    -- Logical Memory Map
```

جدول ث-۴- ادامه

```
        }

END


AddSingleObjectResponse

{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) addSingleObject(4)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN
AddSingleObjectResponse ::= SEQUENCE {
        completionCode          INTEGER {
                                        noError(0),
                                        tagIdNotFound(8),
                                        objectNotAdded(9),
                                        duplicateObject(10),
                                        objectAddedButNotLocked(11),
                                        executionError(255)
                                        },
        executionCode            INTEGER
                                -- See Clause 9.3 and notes in this syntax for a full list of
                                -- executionCodes
    }
END
```

elementNamesهای زیرکه در این پودمان‌ها استفاده شده‌اند، در جای دیگر از این استاندارد تعریف می‌شوند، که به تفصیل شرح داده شده است:

avidodDuplicate (به زیربند ۷-۴-۶ مراجعه کنید)

compactParameter (به زیربند ۷-۳-۵ مراجعه کنید)

completionCode (به زیربند ۹-۲ مراجعه کنید)

executionCode (به زیربند ۹-۳ مراجعه کنید)

object (به زیربند ۷-۳-۴ مراجعه کنید)

objectId (به زیربند ۷-۳-۲ مراجعه کنید)

objectLock (به زیربند ۷-۳-۶ مراجعه کنید)

tagId (به زیربند ۷-۲-۱ مراجعه کنید)

ث-۵ DeleteObjectModules

DeleteObjectModules شامل یک commandModules و responseModule مربوط می‌باشد که به تحقیق‌کننده آموزش می‌دهد که یک **ObjectId** و **Object** آن و پارامترهای مربوطه را حذف کند. تنها یک

برچسب RF و تنها یک **ObjectId** باید به ازای هر فرمان برنامه‌نویسی شود تا مطمئن شویم که فرآیند حذف قوی و مؤثر است. تابع حذف باید **ObjectId**، **Object** مربوطه و پیشرو از Logical Memory Map باید حذف کند.

قواعد نحوی انتزاعی ASN.1 برای DeleteObjectModules در جدول ث-۵- DeleteObjectModules - ارائه می‌شود.

جدول ث-۵- DeleteObjectModules

```
-- Delete Object
-- The DeleteObjectCommand instructs the interrogator to delete the object
-- specified by its OID, from the tag Logical Memory Map. This means that a
-- subsequent command to read the object will return objectNotFound. This
-- procedure might not succeed if the object is locked, if this is found to be the
-- case, the response will return the appropriate completionCode. If the
-- checkDuplicate flag is set to TRUE, the interrogator shall verify, before
-- deleting the requested object, that there is only a single object with the
-- requested OID. If the interrogator detects that several objects have the same
-- OID it shall not perform the DeleteObject function and shall return the
-- appropriate completionCode

-- If the checkDuplicate flag is set to FALSE, the interrogator shall delete the
-- first occurrence of the object specified by its OID.
-- NOTE: This is an argument that effectively provides no protection against
-- duplicate OIDs. It should only be used when there is a high expectation of
-- no duplicates.

DeleteObjectCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) deleteObject(5)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

DeleteObjectCommand ::= SEQUENCE {
        TagId                   OCTET STRING(SIZE(0..255)),
                                -- See Clause 7.2.1 for detailed specification
        objectId                OBJECT IDENTIFIER, -- Full OID value
                                -- This initiates the deletion of the ObjectId and the
                                -- associated Object
        checkDuplicate           BOOLEAN
                                -- If set to TRUE, the interrogator shall check that there is
                                -- only one occurrence of the ObjectId
        }

END
```

جدول ث-۵- ادامه

```
DeleteObjectResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) deleteObject(5)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

DeleteObjectResponse ::= SEQUENCE {
        completionCode                  INTEGER {
                                                noError(0),
                                                tagIdNotFound(8),
                                                duplicateObject(10),
                                                objectNotDeleted(12),
                                                objectIdNotFound(13),
                                                objectLockedCouldNotDelete(14),
                                                executionError(255)
                                                },
        executionCode   INTEGER
                        -- See Clause 9.3 and notes in this syntax for a full list of
                        -- executionCodes
    }
END
```

ث-۶ ModifyObjectModule

ModifyObjectModule شامل یک commandModule و responseModule مربوط می‌باشد که به تحقیق‌کننده آموزش می‌دهد که سه فرآیند مربوطه را انجام دهد:

۱. خواندن کامل Logical Memory Map از برچسب RF.

۲. حذف **ObjectId**، **Object** ویژه و پیشرو مربوطه. اگر نمونه‌های دو نسخه‌ای پیدا شوند این فرآیند بی‌نتیجه می‌ماند.

۳. نوشتن **ObjectId**، **Object** اصلاح شده و پیشرو بازسازی شده.

Object باید در یک سطح دیگری از Logical Memory Map قرار بگیرد. یک موقعیت مشابهی ممکن است به وجود آید اگر متغیر **ObjectLock** در این فرمان به TRUE تنظیم شود. در هر دو مورد، Data Protocol Processor استاندارد ISO/IEC 15962، فرآیند جابجاسازی را کنترل می‌کند. تنها یک برچسب RF و تنها یک **ObjectId** باید به ازای هر فرمان برنامه‌نویسی شود تا مطمئن شویم که فرآیند اصلاح مؤثر و قوی است.

قواعد نحوی انتزاعی ASN.1 برای Modify Object Module در جدول ث-۶- ModifyObjectModule - ارائه می‌شود.

Modify Object Module -۶- جدول ث

```
-- Modify Object
-- The ModifyObjectCommand instructs the interrogator to modify an object, its OID
-- and associated parameters already on the tag Logical Memory Map. It does this by
-- deleting the specified object and associated parameters and writing the new
-- values. To achieve this, the complete Logical Memory Map shall be read from the
-- tag. Any instances of duplicate ObjectIds shall result in the process being
-- aborted.

-- If the object is already locked, it cannot be modified and the appropriate
-- completion code shall be returned.

-- If the byte string, that represents the modified data when prepared for encoding
-- in the Logical Memory Map, is the same length as its previous encodation, the
-- modified value is generally written to the same positions.

-- If the byte string is shorter than the previous encodation, then an offset shall
-- be encoded.

-- If the byte string is longer than the previous encodation, then it needs to be
-- located in a different area of the Logical Memory Map with this process
-- controlled by the Data Protocol Processor of ISO/IEC 15962.
ModifyObjectCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) modifyObject(6)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

ModifyObjectCommand ::= SEQUENCE {
        TagId                   OCTET STRING(SIZE(0..255)),
                                -- See Clause 7.2.1 for detailed specification
        objectId                 OBJECT IDENTIFIER,-- Full OID value
        object                   OCTET STRING,
        compactParameter         INTEGER {
                                applicationDefined(0),
                                -- The object shall not be processed through the data
                                     -- compaction rules of 15962 and remains unaltered
                                compact(1),
                                -- Compact object as efficiently as possible using 15962
                                     -- compaction rules
                                utf8Data(2)
                                -- Data has been externally transformed from a 16-bit coded
                                     -- character set to a UTF-8 string. The object shall not be
                                     -- processed through the data compaction rules of 15962
                                and
                                     -- remains unaltered
                                }(0..15),
```

جدول ث-۶- ادامه

```
        objectLock                  BOOLEAN
                                -- If TRUE the interrogator shall lock the ObjectId, the
                                -- Object, its compaction scheme and other features in the
                                -- Logical Memory Map
}
END
ModifyObjectResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) modifyObject(6)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

ModifyObjectResponse ::= SEQUENCE {
        completionCode          INTEGER {
                                noError(0),
                                objectLockedCouldNotModify(7),
                                tagIdNotFound(8),
                                duplicateObject(10),
                                objectNotModified(21),
                                objectModifiedButNotLocked(22),
                                executionError(255)
                                },
        executionCode           INTEGER
                                -- See Clause 9.3 and notes in this syntax for a full list of
                                -- executionCodes
    }
END
```

ث-۷ ReadSingleObjectModules

ReadSingleObjectModules شامل یک commandModule و responseModule مربوطه می‌باشد که به تحقیق‌کننده آموزش می‌دهد که یک **ObjectId**، **Object** آن و پارامترهای مربوطه را از برچسب RF در Logical Memory Map بخواند. متغیرهای دستور را می‌توان استفاده کرد تا بررسی کنیم **ObjectId** بر روی برچسب RF دو نسخه‌ای نمی‌شود. تنها یک برچسب RF باید به ازای هر فرمان برنامه‌نویسی شود تا مطمئن شویم که فرآیند خواندن قوی و مؤثر است.

قواعد نحوی انتزاعی ASN.1 برای ReadSingleObjectModules در جدول ث-۷- ReadSingleObjectModules- ارائه می‌شود.

جدول ث-۷- ReadSingleObjectModules

```
-- Read Single Object
-- The ReadSingleObjectCommand instructs the interrogator to read the Object
-- specified by its Object Identifier from the tag specified by its TagId.
-- NOTE: There is a ReadMultipleObjectsCommand.

-- If the checkDuplicate flag is set to FALSE, the interrogator shall return the
-- first Object found having the requested OID without checking for duplicates.

-- If the checkDuplicate flag is set to TRUE, the interrogator shall check for
-- duplicate objects having the requested OID. If more than one object with the
-- requested OID is found, the interrogator shall return the first found Object
-- having the requested OID and indicate the presence of duplicates with
-- appropriate Completion code.


ReadSingleObjectCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) readSingleObject(7)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN
ReadSingleObjectCommand ::= SEQUENCE {
        tagId                   OCTET STRING(SIZE(0..255)),
                                -- See Clause 7.2.1 for detailed specification
        objectId                OBJECT IDENTIFIER,-- Full OID value
        checkDuplicate           BOOLEAN
                                -- If set to TRUE, the interrogator shall check that there is
                                -- only one occurrence of the ObjectId
        }
END


ReadSingleObjectResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) readSingleObject(7)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

ReadSingleObjectResponse ::= SEQUENCE {
completionCode                  INTEGER {
                                                noError(0),
                                                tagIdNotFound(8),
                                                duplicateObject(10),
                                                objectIdNotFound(13),
                                                objectNotRead(15),
                                                executionError(255)
                                                },
```

جدول ث-۷- ادامه

```
        executionCode               INTEGER ,
                                    -- See Clause 9.3 and notes in this syntax for a full list
                                    of
                                    -- executionCodes
        object                      OCTET STRING,
        compactParameter             INTEGER {
                                            applicationDefined(0),
                                            -- The object was not originally encoded
                                            through the data
                                                -- compaction rules of 15962, and is as
                                            sent from the
                                                -- source application and might require
                                            additional
                                                -- processing by the receiving application.
                                            utf8Data(2),
                                            -- Data has been externally transformed from
                                            a 16-bit
                                                -- coded character set to a UTF-8 string.
                                            The object
                                                -- needs to be processed through an
                                            external UTF-8
                                                -- decoder.
                                            de-compactedData(15)
                                            -- The object was originally encoded through
                                            the data
                                                -- compaction rules of 15962 and de-
                                            compacted on this
                                                -- read operation and restored to its
                                            original format.
                                            }(0..15),
        lockStatus                   BOOLEAN
                                    -- If TRUE, object is locked
    }
END
```

ث-۸- ReadSingleObjectModules

ReadObjectIdsModules شامل یک commandModule و responseModule مربوطه می‌باشد و به تحقیق‌کننده آموزش می‌دهد که تمام **ObjectId** را از برچسب RF بخواند. این پودمان را می‌توان از قبل در یک فرمان انتخابی[1] استفاده کرد تا یک **Object** ویژه خوانده شود و یا **ObjectIds** دو نسخه‌ای شناسایی شوند بنابراین یک رویه خانه‌داری را می‌توان فراخوانی کرد. ، اگر برچسب RF در Logical Memory Map هیچ **ObjectIds** ذخیره‌ای نداشته باشد،یک پاسخ معتبر، فهرست **ObjectId** خالی را بر می‌گرداند. تنها یک برچسب

1 - Selective Command

RF باید به ازای هر فرمان برنامه‌نویسی شود تا مطمئن شویم که فرآیند خواندن قوی و مؤثر است. قواعد نحوی انتزاعی ASN.1 برای ReadObjectIdsModules در جدول ث-۸- ReadObjectIdsModules- ارائه می‌شود.

جدول ث-۸- ReadObjectIdsModules

```
-- Read Object Ids
-- The ReadObjectIdsCommand instructs the interrogator to read all Object Ids from
-- the tag specified by its TagId. Objects can then be read individually by the
-- ReadObjectCommand.

-- If there are duplicate OIDs, the interrogator shall return them as multiple
-- occurrences of the OID in the objectIdsFound list.

-- If a tag has no Object IDs, the objectIdsFound will be returned empty; and
-- as such the command will be executed without error.


ReadObjectIdsCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) readObjectIds(8)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

        TagId::= OCTET STRING(SIZE(0..255))
                -- See Clause 7.2.1 for detailed specification

END


ReadObjectIdsResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) readObjectIds(8)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

ReadObjectIdsResponse ::= SEQUENCE {
        completionCode          INTEGER {
                                        noError(0),
                                        tagIdNotFound(8),
                                        executionError(255)
                                        },
        executionCode           INTEGER ,
                                -- See Clause 9.3 and notes in this syntax for a full list of
                                -- executionCodes
        objectIdsFound SEQUENCE OF ObjectId
        }
ObjectId ::= OBJECT IDENTIFIER-- Full OID value

END
```

ث-۹ ReadAllObjectsModules

ReadAllObjectsModules شامل یک commandModule و responseModule مربوط می‌باشد و به تحقیق‌کننده آموزش می‌دهد که تمام Logical Memory Map را از برچسب RF بخواند و به یک روش به‌طورکامل ساختاریافته‌ای به آن پاسخ دهد که تمام نمونه‌های موارد زیر را شناسایی می‌کند: **ObjectId**، **Object**، **CompactParameter** و **Lockstatus**. تنها یک برچسب RF باید به ازای هر فرمان برنامه‌نویسی شود که مطمئن شویم که فرآیند خواندن قوی و مؤثر است.

قواعد نحوی انتزاعی ASN.1 برای ReadAllObjectsModules در جدول ث-۹- ReadAllObjectsModules ارائه می‌شود.

جدول ث-۹- ReadAllObjectsModules

```
-- Read All Objects
-- The ReadAllObjectsCommand instructs the interrogator to read all objects, their
-- ObjectIds and associated parameters from the tag specified by its TagId. The
-- interrogator shall return all objects and their parameters of the tag specified
-- by its TagId. If there are duplicate OIDs, the interrogator shall return them
-- as multiple occurrences of the OID, objects, and associated parameters in the
-- Objects list.


ReadAllObjectsCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) readAllObjects(9)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

        TagId::=        OCTET STRING(SIZE(0..255))
                        -- See Clause 7.2.1 for detailed specification

END


ReadAllObjectsResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) readAllObjects(9)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

ReadAllObjectsResponse ::= SEQUENCE {
        completionCode          INTEGER {
                                        noError(0),
                                        tagIdNotFound(8),
                                        objectsNotRead(16),
                                        executionError(255)
                                        },
```

جدول ث-۹- ادامه

```
        executionCode          INTEGER,
                               -- See Clause 9.3 and notes in this syntax for a full list of
                               -- executionCodes
        objects                SEQUENCE OF SEQUENCE {
        objectId               OBJECT IDENTIFIER,
       object                  OCTET STRING,
        compactParameter        INTEGER {
                                        applicationDefined(0),
                                        -- The object was not originally encoded through the
                                            -- data compaction rules of 15962, and is as sent
                                            -- from the source application and might require

                                            -- additional processing by the receiving
                                            --application.
                                        utf8Data(2),
                                        -- Data has been externally transformed from a 16-bit
                                            -- coded character set to a UTF-8 string. The
                                        object
                                            -- needs to be processed through an external UTF-
                                        8
                                            -- decoder.
                                        de-compactedData(15)
                                        -- The object was originally encoded through the data
                                            -- compaction rules of 15962 and de-compacted
                                        on this
                                            -- read operation and restored to its original
                                            --format.
                                        }(0..15),
        lockStatus                       BOOLEAN
                                        -- If TRUE, object is locked

        }
}
END
```

ث-۱۰ ReadLogicalMemoryMapModules

ReadLogicalMemoryMapModules شامل یک commandModule و responseModule مربوطه می‌باشد که به تحقیق‌کننده آموزش می‌دهد که تمام Logical Memory Map را از برچسب RF بخواند و به یک روش به‌طورکامل ساختارنیافته به آن پاسخ دهد (یعنی با برگرداندن مقادیر بایت کد شده). هیچ پردازشی از طریق Data protocol Processor و به عنوان بخشی از این دستور خواندن، روی نمی‌دهد، بنابراین **ObjectIds**، **Objects**، **Compactparameter** و **lockStatus** را نمی‌توان به‌طورمستقیم شناسایی کرد. همچنین اگر یک ساختار فهرست با **accessMethod** تعریف شود، این باید در پاسخ قرار گیرد، اما نباید از بایت‌های دیگر در

Logical Memory Map تشخیص داده شود. مهمترین کارکرد این فرمان برای اهداف تشخیصی[1] می‌باشد، اما می‌توان برای کارکردهای دیگری استفاده کرد که خواندن محتوای کامل Logical Memory Map لازم می‌باشد. تنها یک برچسب RF باید به ازای هر فرمان برنامه‌نویسی شود تا مطمئن شویم که فرآیند خواندن قوی و مؤثر است.

قواعد نحوی انتزاعی ASN.1 برای ReadLogicalMemoryMapModules در جدول ث-۱۰- ReadLogicalMemoryMapModules- ارائه می‌شود.

جدول ث-۱۰- ReadLogicalMemoryMapModules

```
-- Read Logical Memory Map
-- The ReadLogicalMemoryMapCommand instructs the interrogator to return the whole
-- contents of Logical Memory Map of the tag specified by its TagId. It differs
-- from the ReadAllObjectsCommand in that this transfers the raw byte string to the
-- application, without processing through the Data Protocol Processor. This could
-- be used for diagnostic purposes.


ReadLogicalMemoryMapCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126)
readLogicalMemoryMap(10)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

        TagId::=        OCTET STRING(SIZE(0..255))
                        -- See Clause 7.2.1 for detailed specification

END


ReadLogicalMemoryMapResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127)
readLogicalMemoryMap(10)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

ReadLogicalMemoryMapResponse ::= SEQUENCE {
        completionCode              INTEGER {
                                            noError(0),
                                            tagIdNotFound(8),
                                            readIncomplete(19),
                                            executionError(255)
                                            },
        executionCode               INTEGER,
```

1 - Diagnostic Purposes

جدول ث-۱۰- ادامه

```
                                        -- See Clause 9.3 and notes in this syntax for a
                                    full list of
                                    -- executionCodes
        logicalMemoryMap             OCTET STRING
}
END
```

ث-۱۱ InventoryAndReadObjectsModules

InventoryAndReadObjectsModules شامل یک commandModule و responseModule مربوطه می‌باشد که به تحقیق‌کننده آموزش می‌دهد که یک مجموعه خاصی از برچسب‌های RF موجود در فیلد عملیاتی آن را شناسایی کند و سپس داده‌های ویژه را از هر برچسب RF برگرداند. کارکرد فهرست توسط استفاده از **applicationFamilyId** و **identilyMethod** محدود می‌شود، که به‌طوردقیق در زیربند ۸-۷ توضیح داده شده است.

متغیر فرمان **Objectlist** شامل یک فهرست مشترک **ObjectIds** می‌باشد که از هر برچسب RF خوانده می‌شود که با معیار انتخابی تعریف شده است. اگر **ObjectList** پوچ باشد، سپس تحقیق‌کننده باید تمام **ObjectIds** و داده‌های مربوطه را از هر برچسب RF بازیابی کند. پاسخ برای هر **tagId** شامل یک ترتیبی از: **ObjectId**، **Object**، **compactParameter** و **lockStatus** می‌باشد.

قواعد نحوی انتزاعی ASN.1 برای InventoryAndReadObjectsModules در جدول ث-۱۱- InventoryAndReadObjectsModules - ارائه می‌شود.

جدول ث-۱۱- InventoryAndReadObjectsModules

```
-- Inventory and Read Objects
-- The InventoryAndReadObjectsCommand instructs the interrogator to inventory (i.e.
-- to identify) all tags present in its operating area and to read the objects
-- specified by the OID list for each inventoried tag. Each tag is uniquely
-- identified by its TagId.

-- If the objectIdList is NULL, then the interrogator shall retrieve all objects of
-- all inventoried tags.

-- If there are duplicate OIDs, the interrogator shall return them as multiple
-- occurrences of the OID, objects, and associated parameters in the Objects list.


InventoryAndReadObjectsCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126)
```

```
inventoryAndReadObjects(11)}
DEFINITIONS
```

جدول ث-۱۱- ادامه

```
EXPLICIT TAGS ::=
BEGIN

InventoryAndReadObjectsCommand ::= SEQUENCE {
        applicationFamilyId     ApplicationFamilyId,
        identifyMethod           INTEGER {
                                        inventoryAllTags(0),
                                        inventoryAtLeast(1),
                                        inventoryNoMoreThan(2),
                                        inventoryExactly (3)
                                        }(0..15),
        numberOfTags            INTEGER (0..65535),
        objectIdList            SEQUENCE OF ObjectId
}

-- The ApplicationFamilyId separates fundamentally different types of application
-- data, and possibly particular objectIds.
-- Specifying a hex value xx (where x is a non-zero value) selects only the RF tags
-- that have the required data content.
-- Specifying a hex value 00 selects all the RF tags; this may be an appropriate
-- action to undertake a full inventory.
-- Specifying a hex value 0x, or x0, (where x is a non-zero value) might not be
-- logically sound because the RF tags are from different applications and the x
-- value has different meaning.

-- If the identifyMethod is set to inventoryAllTags, the interrogator shall perform
-- a complete inventory of all tags present in its field of operation. The value of
-- numberOfTags is irrelevant and should be set to zero by the application.

-- If the identifyMethod is set to inventoryAtLeast, the interrogator shall perform
-- an inventory of the tags present in its field of operation and shall return a
-- response only once it has identified a number of tags equal to numberOfTags.

-- If the numberOfTags is set to 1, the Interrogator will wait until the first tag
-- has been detected. This is a mechanism to wait for a tag to enter the
-- interrogator field.

-- If the numberOfTags is set to more than 1, the Interrogator will wait until the
-- specified number of tags has been detected. This may lead to an indefinite wait
-- for the response from the interrogator. It is the responsibility of the
-- application to accommodate this potentiality.

-- If the identifyMethod is set to inventoryNoMoreThan, the interrogator shall
-- initiate an inventory of the tags present in its field of operation and shall
```

```
-- return a response with a number of tags lower or equal to numberOfTags.
-- The interrogator may interrupt the inventory process when the numberOfTags has
-- been reached or may continue the inventory process till all tags have been read.
-- Note: This may be constrained by the air interface and anticollision
-- mechanism.
```

جدول ث-۱۱- ادامه

```
                                        },

ApplicationFamilyId ::= SEQUENCE {
        applicationFamily       INTEGER(0..15),
        applicationSubFamily    INTEGER(0..15)
        }
        -- The code values are defined in the ConfigureAfiCommand.

ObjectId ::= OBJECT IDENTIFIER-- Full OID value

END


InventoryAndReadObjectsResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127)
inventoryAndReadObjects(11)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

InventoryAndReadObjectsResponse ::= SEQUENCE {
        completionCode                  INTEGER {
                                                noError(0),
                                                objectsNotRead(16),
                                                failedToReadMinimumNumberOfTags(23),
                                                -- for example, this could be due to a time-out
                                                failedToReadExactNumberOfTags(24),
                                                -- for example, this could be due to a time-out
executionError(255)
executionCode                   INTEGER ,
                                        -- See Clause 9.3 and notes in this syntax for a full list
                                        of
                                        -- executionCodes
        numberOfTagsFound               INTEGER (1..65535),
        tagIdAndObjects                 SEQUENCE {
                tagId                   OCTET STRING(SIZE(0..255)),
                                        -- See Clause 7.2.1 for detailed specification
        objects SEQUENCE OF SEQUENCE {
                objectId        OBJECT IDENTIFIER,
                object          OCTET STRING,
                                        compactParameter INTEGER {
```

```
                                        applicatioDefined(0),
                                        -- The object was not originally encoded through the
                                            -- data compaction rules of 15962, and is as sent
                                            -- from the source application and might require
END
```

جدول ث-۱۱- ادامه

```
                                            - additional processing by the receiving
                                        application.
                                        utf8Data(2),
                                        -- Data has been externally transformed from a 16-bit
                                            -- coded character set to a UTF-8 string.The
                                        object
                                            -- needs to be processed through an external UTF-
                                        8
                                        -- decoder.
                                        de-compactedData(15)
                                        -- The object was originally encoded through the data
                                            -- compaction rules of 15962 and de-compacted
                                        on this
                                            -- read operation and restored to its original
                                            -- format.
                                        }(0..15),
        lockStatus                       BOOLEAN
                                        -- If TRUE, object is locked
        }
    }
}
```

ث-۱۲ پاک کردن پودمان‌ها حافظه

EraseMemoryModules شامل یک commandModule و responseModule مربوطه می‌باشد که به تحقیق‌کننده آموزش می‌دهد که تمام Logical Memory Map از برچسب ویژه RF، را با مقدار صفر دوباره راه‌اندازی کند. اگر به عنوان **accessMethod** تعریف شود، شامل فهرست راهنما می‌باشد. اگر هیچ‌کدام از این بلوک‌ها قفل نشود، این باید باعث حذف تمام **ObjectId**، **Objects** و پیشروهای مربوطه شود. اگر همه بلوک‌ها قفل شوند، آنگاه **CompletionCode**:blocksLocked برگشت داده می‌شود. **objects** که روی برچسب RF می‌مانند می‌توانند با استفاده بعدی از ReadObjectIdsModule شناسایی شوند (به پیوست 0 مراجعه کنید). تنها یک برچسب RF باید به ازای هر فرمان برنامه‌نویسی شود تا مطمئن شویم که پاک کردن فرآیند قوی و مؤثر است.

قواعد نحوی انتزاعی ASN.1 برای EraseMemoryModules درجدول ث-۱۲- EraseMemoryModules- ارائه می‌شود.

جدول ث-۱۲- EraseMemoryModules

```
-- Erase Memory
-- The EraseMemoryCommand instructs the interrogator to reset to zero the whole
-- Logical Memory Map of the tag specified by its TagId. This results in the
-- deletion of all ObjectIds, Objects and associated parameters. The objects that
-- are locked cannot be deleted. If some objects cannot be deleted because they are
-- locked, the interrogator shall return the appropriate completionCode. Remaining
-- Objects can then be identified by the ReadObjectIdsCommand.


EraseMemoryCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) eraseMemory(12)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

        TagId::=                OCTET STRING(SIZE(0..255))
                                -- See Clause 7.2.1 for detailed specification

END


EraseMemoryResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) eraseMemory(12)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

EraseMemoryResponse ::= SEQUENCE {
        completionCode          INTEGER {
                                        noError(0),
                                        tagIdNotFound(8),
                                        blocksLocked(17),
                                        eraseIncomplete(18),
                                        -- process incomplete for some undefined reason, but it
                                            -- is possible to re-invoke the command to complete
                                        executionError(255)
                                        },
        executionCode           INTEGER
                                -- See Clause 9.3 and notes in this syntax for a full list of
                                -- executionCodes
        }
END
```

ث-۱۳ GetApplication-basedSystemInformationModules

GetApplication-basedSystemInformationModules شامل یک CommandModule و ResponseModule مربوطه می‌باشد که به تحقیق‌کننده آموزش می‌دهد که اطلاعات سامانه را بخواند و متغیرهایی را برگرداند که به برنامه کاربردی مربوط می‌شوند، یعنی applicationFamilyId و storageFormat. قواعد نحوی انتزاعی ASN.1 برای GetApplication-basedSystemInformationModules در جدول ث-۱۳- GetApplication-basedSystemInformationModules- ارائه می‌شود.

جدول ث-۱۳- GetApplication-basedSystemInformationModules

```
-- Get Application-based System information
-- The GetApp-basedSystemInfoCommand instructs the interrogator to read the System
-- information from the tag specified by its TagId, and abstract the parameters
-- relevant to the application data: the applicationFamilyId and the storageFormat.


GetApp-basedSystemInfoCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) getApp-basedSystemInfo(13)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

        TagId::=                OCTET STRING(SIZE(0..255))
                                -- See Clause 7.2.1 for detailed specification

END


GetApp-basedSystemInfoResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) getApp-basedSystemInfo(13)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

GetApp-basedSystemInfoResponse ::= SEQUENCE {
        completionCode              INTEGER {
                                            noError(0),
                                            tagIdNotFound(8),
                                            systemInfoNotRead(20),
                                            executionError(255)
                                            },
        executionCode               INTEGER,
                                    -- See Clause 9.3 and notes in this syntax for a full
                                    -- list of executionCodes
        applicationFamilyId         ApplicationFamilyId,
        storageFormat               StorageFormat
```

جدول ث-۱۳- ادامه

```
        }

        ApplicationFamilyId ::= SEQUENCE {
        applicationFamily INTEGER(0..15),
        applicationSubFamily INTEGER(0..15)
        }
        -- The code values are defined in the ConfigureAfiCommand.

StorageFormat ::= SEQUENCE {
        accessMethod INTEGER (0..3),
        dataFormat INTEGER (0..31)
        }
        -- The code values are defined in the ConfigureStorageFormatCommand.

END
```

ث-۱۴ افزایش چند پودمان اشیا

AddMultipleObjectsModules شامل یک commandModule و responseModule مربوطه می‌باشد که به تحقیق‌کننده آموزش می‌دهد که یک مجموعه‌ای از **objectIds** آن‌ها و پارامترهای مربوطه را در برچسب RF در Logical Memory Map بخواند. متغیرهای این فرمان را می‌توان برای قفل کردن هر یک از **ObjectId**، **Object** و پارامترهای مربوطه استفاده کرد، و همچنین برای بررسی کردن اینکه **ObjectId** بر روی برچسب RF هنوز کدبندی نشده، نیز می‌توان استفاده کرد. تنها برچسب RF باید به ازای هر فرمان برنامه‌نویسی شود تا مطمئن شویم که فرآیند نوشتن قوی و مؤثر است.

قواعد نحوی انتزاعی ANS.1 برای AddMultipleObjectsModules در جدول ث-۱-AddMultipleObjectsModules- ارائه می‌شود.

جدول ث-۱۴- AddMultipleObjectsModules

```
-- Add Multiple Objects
-- The AddMultipleObjectsCommand instructs the interrogator to write a sequence of
-- objects, their ObjectIds and associated parameters into the tag logical memory
-- map.
-- NOTE: There is also an AddSingleObjectCommand.

-- If the checkDuplicate flag is set to TRUE, the interrogator shall verify, before
-- adding the objects, that no object with the same OID already exists. If such an
-- object exists, the interrogator shall not perform the Add Object function and
-- shall return the appropriate Completion Code.

-- If the Lock flag is set to TRUE, the interrogator shall lock the ObjectId, the
-- Object, its compaction scheme and associated parameters into the tag Logical
-- Memory Map
```

جدول ث-۱۴- ادامه

```
AddMultipleObjectsCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) addMultipleObjects(14)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN
AddMultipleObjectsCommand ::= SEQUENCE {
tagId                                   OCTET STRING(SIZE(0..255)),
                                                -- See Clause 7.2.1 for detailed specification
        addObjectsList                          SEQUENCE OF SEQUENCE{
        objectId OBJECT                          IDENTIFIER,-- Full OID value
        avoidDuplicate                           BOOLEAN,
                                                -- If set to TRUE, check for duplicate objectId
        object                                   OCTET STRING,
        compactParameter                         INTEGER {
                                                        applicationDefined(0),
                                                        -- The object shall not be processed through the data
                                                            -- compaction rules of 15962 and remains unaltered
                                                        compact(1),
                                                        -- Compact object as efficiently as possible using
                                                            -- 15962 compaction rules
                                                        utf8Data(2)
                                                        -- Data has been externally transformed from a 16-bit
                                                            -- coded character set to a UTF-8 string. The object
                                                            -- shall not be processed through the data
                                                        compaction
                                                            -- rules of 15962 and remains unaltered
                                                        }(0..15),
        objectLock                              BOOLEAN
                                                -- If TRUE the interrogator shall lock the ObjectId, the

                                                    -- Object, its compaction scheme and other features in
                                                    -- the Logical Memory Map
                }
        }
END

AddMultipleObjectsResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) addMultipleObjects(14)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

AddMultipleObjectsResponse ::= SEQUENCE {
        tagWriteResponse                        SEQUENCE OF SEQUENCE{
        objectId                                 OBJECT IDENTIFIER,-- Full OID value
        completionCode                           INTEGER{
                                                        noError(0),
                                                        tagIdNotFound(8),
```

جدول ث-۱۴- ادامه

```
                                        objectNotAdded(9),
                                        duplicateObject(10),
                                        objectAddedButNotLocked(11),
                                        executionError(255)
                                        }
            },
    executionCode               INTEGER
                                -- See Clause 9.3 and notes in this syntax for a full list of
                                -- executionCodes
    }
                        END -- executionCodes
    } END
```

ث-۱۵ خواندن چند پودمان اشیا

ReadMultipleObjectsModules شامل یک commandModule و responseModule مربوطه می‌باشد که به تحقیق‌کننده آموزش می‌دهد که یک مجموعه‌ای از **objectIds** و **objects** آن‌ها و پارامترهای مربوطه را از برچسب RF در Logical Memory Map بخواند. متغیرهای Command را می‌توان استفاده کرد تا بررسی کنیم که **objectId** بر روی برچسب RF دو نسخه‌ای نمی‌شود. تنها یک برچسب RF باید به ازای هر فرمان برنامه‌نویسی شود تا مطمئن شویم که فرآیند خواندن قوی و مؤثر است.

قواعد نحوی انتزاعی ASN.1 برای ReadMultipleObjectsModules در جدول ث-۱۵ -ReadMultipleObjectsModules ارائه می‌شود.

جدول ث-۱۵- ReadMultipleObjectsModules

```
-- Read Multiple Objects
-- The ReadMultipleObjectsCommand instructs the interrogator to read the Objects
-- specified by their Object Identifiers from the tag specified by its TagId.
-- NOTE: There is a ReadSingleObjectCommand.

-- If the checkDuplicate flag is set to FALSE, the interrogator shall return the
-- first Object found having the requested OID without checking for duplicates.
-- If the checkDuplicate flag is set to TRUE, the interrogator shall check for
-- duplicate objects having the requested OID. If more than one object with the
-- requested OID is found, the interrogator shall return the first found Object
-- having the requested OID and indicate the presence of duplicates with
-- appropriate Completion code.


ReadMultipleObjectsCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) readMultipleObjects(15)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN
```

جدول ث-۱۵- ادامه

```
        ReadMultipleObjectsCommand ::= SEQUENCE {
        tagId                           OCTET STRING(SIZE(0..255)),
                                        -- See Clause 7.2.1 for detailed specification
        readObjectList                  SEQUENCE OF SEQUENCE{
        objectId                        OBJECT IDENTIFIER,-- Full OID value
        checkDuplicate                  BOOLEAN
                                        -- If set to TRUE, the interrogator shall check that
                                        -- there is only one occurrence of the ObjectId
        }
    }
END

ReadMultipleObjectsResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) readMultipleObjects(15)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

ReadMultipleObjectsResponse ::= SEQUENCE {
        tagReadResponse                 SEQUENCE OF SEQUENCE{
        objectId                        OBJECT IDENTIFIER,
        object                          OCTET STRING,
        compactParameter                INTEGER {
                                                applicationDefined(0),
                                                -- The object was not originally encoded through the
                                                    -- data compaction rules of 15962, and is as sent
                                                    -- from the source application and might require
                                                    -- additional processing by the receiving
                                                    -- application.
                                                    utf8Data(2),
                                                -- Data has been externally transformed from a 16-bit
                                                    -- coded character set to a UTF-8 string. The object
                                                    -- needs to be processed through an external UTF-8
                                                -- decoder.
                                                de-compactedData(15)
                                                -- The object was originally encoded through the data
                                                    -- compaction rules of 15962 and de-compacted on
                                                this
                                                    -- read operation and restored to its original
                                                    -- format.
                                                }(0..15),
                lockStatus              BOOLEAN,
                                        -- If TRUE, object is locked
                completionCode          INTEGER {
                                                noError(0),
                                                tagIdNotFound(8),
                                                duplicateObject(10),
                                                objectIdNotFound(13),
```

جدول ث-۱۵- ادامه

```
                                        objectNotRead(15),
                                        executionError(255)
                                        }
        },
executionCode           INTEGER
                        -- See Clause 9.3 and notes in this syntax for a full list of
                        -- executionCodes
} END
```

ث-۱۶ ReadFirstObjectModules

ReadFirstObjectModules شامل یک commandModule و responseModule مربوطه می‌باشد که به تحقیق‌کننده آموزش می‌دهد که یک شی در اولین موقعیت بر روی برچسب RF بخواند تا **objectId**، **Object** آن و پارامترهای مربوطه را از برچسب RF بر روی Logical Memory Map برگشت دهد. تنها یک برچسب RF باید به ازای هر فرمان برنامه‌نویسی شود تا مطمئن شویم که این فرآیند خواندن قوی و مؤثر است.

یک فرآیند مؤثر را می‌توان با توانایی انتقال فقط بلوک‌های مناسب داده‌ها از میان واسط هوایی به دست آورد. این نیاز به برنامه کاربردی دارد تا موارد زیر را فراهم کند:

- چند درجه اطمینان که **Object** مورد نظر در موقعیت ویژه کدبندی می‌شود، که می‌توان با قواعدی در استاندارد کاربردی به دست آورد.
- طول مشخص یا حداکثر طول فشرده **Object** مورد نظر. این برای مشخص کردن تعداد هشتایی‌ها که انتقال داده می‌شود، استفاده می‌شود و در نتیجه **Object** مورد انتظار بر گردانده می‌شود. مقدار طول فشرده، را می‌توان با رویه‌ای تعیین کرد که در قسمت زیر شرح داده شده است.
- مقدار **ObjectId** مورد انتظار برای فعال کردن Data Protocol Processor که این را به هشتایی‌های کدبندی شده تبدیل می‌کند و پارامترهای کد شده دیگر را به طول فشرده اضافه می‌کند.

طرح‌های فشرده‌سازی و نویسههای مناسبی که می‌توانند فشرده شوند در جدول ث-۱۶- طرح‌های فشرده‌سازی و نویسهها- تعریف می‌شوند.

جدول ث-۱۶- طرح‌های فشرده‌سازی و نویسهها

طرح‌های فشرده‌سازی	تعداد نویسه‌ها	نویسه‌هایی که پشتیبانی می‌شوند
عددی (۴ بیت)	۱۰	0..9
۵ بیت	۳۱	A..Z [\ ] ^ _
۶ بیت	۶۴	همانند عددی + ۵ بیت و نویسه‌های زیر: SPACE ! " # $ % & ' ( ) * + , - . / : ; < = > ? @
۷ بیت	۱۲۸	همه نویسه‌های استاندارد ISO/IEC 646 شامل نویسه‌های کنترلی
۸ بیت	۲۵۶	همه نویسه‌های استاندارد ISO/IEC 8859-1 شامل نویسه‌های کنترلی

به عنوان یک مثال از کاربرد آن، یک استاندارد کاربردی را در نظر بگیرید که برای واحدهای انتقال استاندارد ISO/IEC15459 به Unique Item Identifier نیاز دارد که در اولین موقعیت واقع می‌شود. حداکثر طول این شناسه ۳۵ نویسههای الفبایی عددی[1] می‌باشد. فرض کنید که نقطه ضبط[2] داده‌های ویژه، واحدهای حمل و نقل را از منابع مختلف کنترل می‌کند و هیچ دانش قبلی از طول کدهای ویژه ندارد، بنابراین بدترین مورد باید ایجاد شود. جدول ث-۱۶- طرح‌ها و نویسههای فشرده‌سازی - فهرستی از نویسههایی را تهیه می‌کند که می‌توانند با طرح‌های فشرده‌سازی استاندارد ISO/IEC 15962 کدبندی شوند. نویسههای الفبایی عددی که توسط استاندارد ISO/IEC 15459 مشخص شده‌اند می‌توانند با طرح ۶ بیتی فشرده شوند. بنابراین حداکثر طول شی کد شده $(35\times 6/8 = 26/25)$ ۲۷ هشتایی است، زیرا این مقدار باید به بالا گرد شود[3] تا به درستی کدبندی شود. Data Protocol Processor به این طول کد شده **ObjectId** و پارامترهای مربوطه اضافه می‌کند تا تعداد بلوک‌هایی که منتقل می‌شوند را تعیین کند.

اگر شی، طول و ساختار مشخصی داشته باشد، یعنی یک مقدار کد ۲۰ رقمی، جدول نشان می‌دهد که می‌تواند با استفاده از طرح فشرده‌سازی عددی ۴ بیت در هر رقم، فشرده شود. بنابراین طول فشرده کلی **Object**، ۱۰ هشتایی است. اگرچه فشرده‌سازی عدد صحیح ممکن است برای تولید یک **Object** کد شده کوتاه‌تر فراخوانی شود، محاسبه طول براساس فشرده‌سازی عددی ساده‌تر، تمام موارد برای Unique Item Identifier عددی را پوشش می‌دهد.

هدف اصلی، به حداقل رساندن انتقال از میان واسط هوایی با درجه بالای اطمینان می‌باشد که شی بر گردانده می‌شود. می‌توان این فرآیند را اصلاح کرد تا تعداد هشتایی‌ها تعیین شوند. طول **Object** مورد انتظار را می‌توان با شبیه‌سازی فرآیند فشرده‌سازی با استفاده از نمونه‌های اشیا واقعی برگشتی دقیق‌تر محاسبه کرد.

قواعد نحوی انتزاعی ASN.1 برای ReadFirstObjectModules در جدول ث-۱۷- ReadFirstObjectModules- ارائه می‌شود.

جدول ث-۱۷- ReadFirstObjectModules

```
-- Read First Object
-- The ReadFirstCommand instructs the interrogator to read the object specified
-- by its position stored in the tag.

-- This command might have features across the air interface that could be faster
-- than reading a single named object. The application may therefore select to have
-- the most often accessed object stored first in the tag.

ReadFirstObjectcommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) readFirstObject(16)}
DEFINITIONS
```

1 - Alphanumeric
2 - Capture
3 - Rounded up

جدول ث-۱۷- ادامه

```
EXPLICIT TAGS ::=
BEGIN

ReadFirstObjectCommand ::= SEQUENCE {
        TagId               OCTET STRING(SIZE(0..255)),
                            -- See Clause 7.2.1 for detailed specification
        objectId            OBJECT IDENTIFIER, -- This is the expected objectId.
                            -- Its value is used by the processes of ISO/IEC 15962 to
                            -- determine the number of blocks that need to be transferred. It
                            -- is not used as part of the search process itself
        maxAppLength  INTEGER(1..65535)
                            -- This value is specified by the application to represent the
                            -- maximum compacted length of the expected object
}

END

ReadFirstObjectResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) readFirstObject(16)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

ReadFirstObjectResponse ::= SEQUENCE {
        ObjectId                OBJECT IDENTIFIER, -- This is the actual ObjectId as found in
                                -- the first position
        object                   OCTET STRING,
        compactParameter        INTEGER {
                                                        applicationDefined(0),
                                        -- The object was not originally encoded through the data
                                         -- compaction rules of 15962, and is as sent from the
                                         -- source application and might require additional
                                         -- processing by the receiving application.
                                          utf8Data(2),
                                         -- Data has been externally transformed from a 16-bit
                                         -- coded character set to a UTF-8 string. The object
                                         -- needs to be processed through an external UTF-8
                                         -- decoder.
                                       de-compactedData(15)
                                      -- The object was originally encoded through the data
                                         -- compaction rules of 15962 and de-compacted on this

                                         -- read operation and restored to its original format.
                                      }(0..15),
        lockStatus                BOOLEAN,
                                -- If TRUE, object is locked
        completionCode           INTEGER {
```

جدول ث-۱۷- ادامه

```
                                        noError(0),
                                        tagIdNotFound(8),
                                        objectIdNotFound(13),
                                        objectNotRead(15),
                                        executionError(255)
                                        },
        executionCode           INTEGER
                                -- See Clause 9.3 and notes in this syntax for a full list of
                                -- executionCodes
        }
END
```

پیوست ج
(اطلاعاتی)
مثالی از یک کدبندی انتقال در استاندارد ISO/IEC15961: 2004

این پیوست قواعد کدبندی اصلی مشخص شده را در این استاندارد ISO/IEC15961: 2004 با نشان‌دادن نمونه‌ای در هشتایی‌های یک دستور و پاسخ AddMultipleObjects (فرضی) شرح می‌دهد.

ج-۱ توصیف کارکردی فرمان

کارکرد فرمان شرح داده می‌شود که نوشتن دو شی‌داده در برچسب RFID می‌باشد که شناسه اقلام منحصربه‌فرد C73779C2B7A3DBEF دارد. جزئیات دیگر به شرح زیر می‌باشند:

- اولین Object Identifier، "30 10 15691 0 1" با شی "ABC 123456" که قفل می‌شود.
- دومینObject Identifier، "17 10 15691 0 1" با شی "50" که قفل نمی‌شود.
- هیچ شرطی وجود ندارد که از دو نسخه‌ای شدن داده‌ها که بر روی برچسب RFID کدبندی می‌شود، جلوگیری کند.
- هر دو شی فشرده می‌شوند.

ج-۲ قواعد نحوی انتزاعی برای فرمان AddMultipleObjects

این در پیوست ث-۱۴ نشان داده می‌شود.

ج-۳ فرمان AddMultipleObjects با مقادیر داده

```
AddMultipleObjectsCommand   -- {1 0 15961 126 14}
    ::=                     {
     tagId                  C7 37 79 C2 B7 A3 DB EF 16
     addObjectsList         {

     -- 1st object
     {
     objectId               {1  0  15961  10  30},
     avoidDuplicate         FALSE,
     object                 “ABC123456”,
     compactParameter       compact(1),
     objectLock             TRUE
     },

     -- 2nd object
     {
     objectId               {1  0  15961  10  17},
     avoidDuplicate         FALSE,
     object                 “50”,
     compactParameter       compact(1),
     objectLock             FALSE
     }
```

}

}

یادآوری- مقادیر ObjectId به صورت OBJECT IDENTIFIERS نشان داده می‌شوند. مقادیر شی در "" مانند نویسههای قابل چاپ نشان داده می‌شوند.

ج-۴ کدبندی انتقال برای فرمان مثال

کدبندی انتقال در هشتایی مقدار فرمان در بالا ارائه شده‌اند (پس از به کارگیری قواعد کدبندی انتقال که در استاندارد ISO/IEC 15961: 2004 تعریف شد)، در زیر به شکل جدولی نشان داده می‌شود. هر ردیف نشان دهنده یکی از خطوط پیوست ج-۳ است. مقادیر Type identifiers از طول‌ها و از مضمون‌ها در شانزده شانزدهی نشان داده می‌شوند، دو رقم شانزده شانزدهی در هر هشتایی.

قواعد نحوی انتزاعی	نوع	طول	مقدار
AddMultipleObjectsCommand	06	05	28 FC 59 7E 0E
SEQUENCE	30	3F	
TagId	04	08	C7 37 79 C2 B7 A3 DB EF
SEQUENCE OF (AddObjectsList)	30	33	
SEQUENCE	30	1B	
ObjectId	06	05	28 FC 59 0A 1E
AvoidDuplicate	01	01	00
Object	04	09	41 42 43 31 32 33 34 35 36
CompactParameter	02	01	01
ObjectLock	01	01	FF
SEQUENCE	30	14	
ObjectId	06	05	28 FC 59 0A 11
AvoidDuplicate	01	01	00
Object	04	02	25 30
CompactParameter	02	01	01
ObjectLock	01	01	00

توضیح بیشتر در مورد مقادیر کد شده ویژه در زیر ارائه می‌شوند، که تعداد، نشان دهنده تعداد در فهرست جدولی می‌باشد.

۱. مقادیر در ستون **Type** از قواعد زیربند ۶-۲-۲ از استاندارد ISO/IEC 15961: 2004 حاصل می‌شوند.
۲. پودمان OBJECT IDENTIFIER با استفاده از همان قواعد برای OBJECT IDENTIFIER کدبندی می‌شود.
۳. همان‌طور که در زیربند ۶-۲-۹ از استاندارد ISO/IEC 15961: 2004 تعریف شده SEQUENCE هیچ مقدار مرتبط ندارد.

۴. همان‌طور که در زیربند ۶-۲-۱۰ از استاندارد ISO/IEC 15961: 2004 تعریف شده است، متغیر **addObjectsList** یک SEGUENCE OF را مشخص می‌کند که هیچ مقدار مرتبطی ندارد.
۵. زمانیکه مجموعه‌های objcectId و داده‌های مرتبط وجود دارند، طول تمام کدبندی تودرتو باید روی خط کدبندی شود.
۶. متغیرهای **avoidDuplicate** و **objectLock**، BOOLEAN هستند. مقدار کدبندی شده برای BOOLEAN = FALSE باید **00** باشد. مقدار کدبندی شده برای BOOLEAN = TRUE ممکن است هر مقداری غیر از **00** باشد.
۷. همان‌طور که در زیربند ۷-۳-۳ از استاندارد ISO/IEC 15961: 2004 تعریف شده است، متغیر **compactParameter** همانند یک INTEGER کدبندی می‌شود.

کدبندی انتقال کامل دستور همانند یک جریان هشتایی است:

06 05 28 FC 59 7E 0E 30 3F 04 08 C7 37 79 C2 B7 A3 DB EF 30 33 30 1B 06 05 28 FC 59 0A 1E 01 01 00 04 09 41 42 43 31 32 33 34 35 36 02 01 01 01 01 FF 30 14 06 05 28 FC 59 0A 11 01 01 00 04 02 35 30 02 01 01 01 01 00.

ج-۴ توصیف کارکردی پاسخ

برای این مثال، فرض کنید که برچسب RFID پیدا شده است و هر دو مجموعه Object Identifier و Object به درستی به برچسب RFID اضافه شده‌اند و از تمام فرآیندهای صحیح استاندارد ISO/IEC 15962 استفاده می‌شود، با یک مورد استثنا مهم که اولین شی هنگامی که توسط فرمان فراخوانی می‌شود، نمی‌تواند قفل شود. زمانی که این فرمان با موفقیت اجرا شد، برحسب ارتباطات سامانه‌ها، executionCodeایی که برگردانده می‌شود "0 noError" می‌باشد.

ج-۶ قواعد نحوی انتزاعی برای پاسخ AddMultipleObjects

این در پیوست ث-۱۴ نشان داده می‌شود.

ج-۷ پاسخ AddMultipleObjects با مقادیر داده‌ها

```
AddMultipleObjectsResponse -- { 1 0 15961 127 14 }
        ::= {
      tagWriteResponse {
        -- object 1 add response
        {
        objectId                { 1 0 15961 10 30 },
        completionCode        11
        },
        -- object 2 add response
        {
        objectId { 1 0 15961 10 17 },
        completionCode 0
        }
executionCode 0
```

}

ج-۸ کدبندی انتقال برای پاسخ مثال

کدبندی انتقال در هشتایی‌های مقدار پاسخ که در بالا ارائه شده است (پس از به کارگیری قواعد کدبندی اصلی تعریف شده در این استاندارد)، در قسمت زیر به صورت جدولی نشان داده می‌شود که هر ردیف نشان دهنده یکی از خطوط پیوست ج-۷ می‌باشد. مقادیر Type identifiers، طول‌ها و محتویات، در شانزده شانزدهی نشان داده می‌شوند، دو رقم شانزده شانزدهمی در هر هشتایی.

قواعد نحوی انتزاعی	نوع	طول	مقدار
AddMultipleObjectsResponse	06	05	28 FC 59 7F 0F
SEQUENCE	30	1D	
SEQUENCE OF (TagWriteResponse)	30	18	
SEQUENCE	30	0A	
ObjectId	06	05	28 FC 59 0A 1E
CompletionCod	02	01	0B
SEQUENCE	30	0A	
ObjectId	06	05	28 FC 59 0A 11
CompletionCode	02	01	00
ExecutionCode	02	01	00

کدبندی انتقال کامل پاسخ به عنوان یک جریان هشتایی عبارتست از.

06 05 28 FC 59 7F 0F 30 1D 30 18 30 0A 06 05 28 FC 59 0A 1E 02 01 0B 30 0A 06 05 28 FC 59 0A 11 02 01 00 02 01 00

کتابنامه

[1] ISO/IEC 8824-1, Information technology — Abstract Syntax Notation One (ASN.1) — Specification of basic notation (equivalent to ITU-T Recommendation X.680)

[2] ISO/IEC 8825-1, Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) (equivalent to ITU-T Recommendation X.690)

[3] ISO/IEC 15961-2, Information technology — Radio frequency identification (RFID) for item management: Data protocol — Part 2: Registration of RFID data constructs

[4] ISO/IEC 15962:2004, Information technology — Radio frequency identification (RFID) for item management — Data protocol: data encoding rules and logical memory functions

[5] ISO/IEC 18000-2, Information technology — Radio frequency identification for item management — Part 2: Parameters for air interface communications below 135 kHz

[6] ISO/IEC 18000-3, Information technology — Radio frequency identification for item management — Part 3: Parameters for air interface communications at 13,56 MHz

[7] ISO/IEC 18000-4, Information technology — Radio frequency identification for item management — Part 4: Parameters for air interface communications at 2,45 GHz

[8] ISO/IEC 18000-6, Information technology — Radio frequency identification for item management — Part 6: Parameters for air interface communications at 860 MHz to 960 MHz

[9] EPCglobal Tag Data Standards version 1.4

[10] RFC 3061, A URN Namespace of Object Identifiers, published by The Internet Engineering Task Force, part of The Internet Society (2001)