**"בוב" –**

**מסמך אפיון וניתוח**

**מגישים: אייל לוי**

**שאול רויזן**

**תאריך: אפריל 2014**

**ניסן תשע"ד**

**מסמך אפיון וניתוח**

**מטרת הפרויקט:**

אפליקציה מתקדמת לעריכת ויצירת קבצי XML באמצעות כלים ויזואליים.

הלקוחות הינם אנשי חומרה או משתמשים שאינם מתכנתים, הרוצים להגדיר מפרט היררכי המתאר תקשורת בין תוכנות או תוכנה וחומרה. המערכת תאפשר יצירה ועריכה של הקבצים הללו, בהתאם לסכמה שתקבל, כך שהמשתמש לא יצטרך להתעסק עם עורך הטקסט, אלא עם תצוגה ויזואלית בלבד, כולל חיבור לוגי בין טיפוסים שונים במפרט. המערכת תבטיח את נכונות הקובץ על פי הסכמה, תסתיר מהמשתמש עבודה מול הקובץ ואת ההיכרות הפנימית עם מבנה ה XML.

בנוסף, המערכת תאפשר הרחבה של הסכמה לדרישות מיוחדות המקשרות בין טיפוסים שונים וגם מוסיפה Meta Data שימושי עבור תוכנות שעבורן ה XML מהווה קובץ קלט. ההרחבות לא מוגדרות על פי הסכמה, אך מקובלות בעולם מפרטי תקשורת.

ישנה סכמה עיקרית, בשם copperhead.xsd בשבילה נועדה התוכנה, ולגביה יינתנו הדוגמאות, אך ניתן להשתמש במערכת עבור כל סכמה אחרת באותה צורה.

**המצב כיום:**

כותבי מפרטי תקשורת צריכים כיום לעשות זאת ידנית, עם Notepad או עורך XML כלשהו, כאשר הם צריכים להבין את החוקיות וסוגי הטיפוסים לשימוש בעל פה.

ה XSD נמצא כחלק מהכלים שעבורם המפרט מהווה קובץ קלט ומוודאים אותו, אך יצירת הקובץ עצמה היא ידע העובר במסמכים ובעל פה במקום שימוש בסכמה והסטנדרטים המוגדרים בה. העורך מוסיף טיפוס אחר טיפוס, מבצע את הקישורים בעצמו ולאחר מכן משתמש בכלי שבודק האם יש שגיאות. התהליך ארוך מכיוון שבמהלך הכתיבה מתרחשות טעויות כתיב או בגלל חוסר הבנה של המבנה החוקי.

בנוסף, ישנן אפשרויות שהעורך לא זוכר שקיימות, לא מודע אליהן, מה שנותן תוצר סופי נמוך באיכותו, ומצריך מעבר של אדם אחר, מנוסה יותר, כדי להגיע לרמה מספקת.

**משתמשי המערכת:**

*מנהל המערכת*

מנהל המערכת אחראי על הפעילות התקינה של המערכת ותוצריה.

מלבד הפעולה הבסיסית של כניסה למערכת, הוא היחיד שיכול לבצע את הפעולות הבאות:

* החלפת סכימה – זוהי פעולה שמשפיעה על התנהגות התוכנה ועל סוגי המפרטים שהתוכנה מאפשרת לכתוב.
* הצגת דו"חות סיכום על כלל פעילות המערכת.

*צופה*

משתמש שרק צופה בהיררכיות המפרטים ועושה חיתוכים וחיפושים בתוכו כדי לדעת איך התקשורת עובדת, לבחון את הטיפוסים השונים ולוודא שהמפרט שבידו הוא תקין. להלן רשימת הפעולות:

* כניסה למערכת.
* חיפוש טיפוס וביצוע חיתוכים על טיפוסים שונים.
* צפייה במפרט קיים.
* בדיקת תקינות של מפרט שקיבל, כדי לוודא שאפשר להתחיל לעבוד איתו.

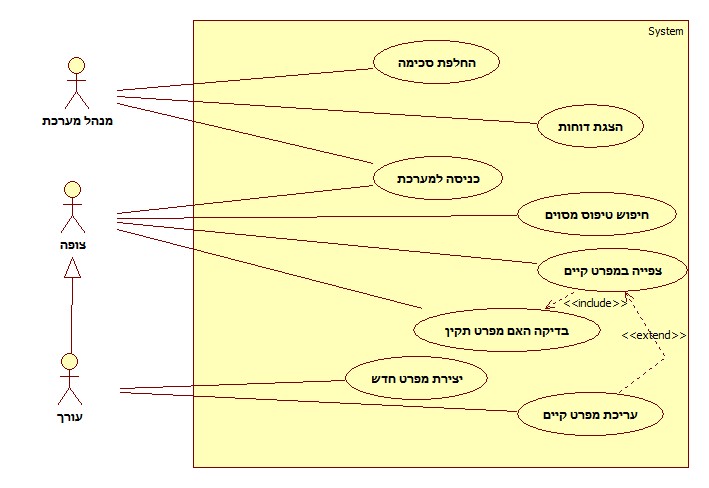
*עורך*

זהו המשתמש העיקרי של המערכת. העורך אחראי ליצור את המפרטים דרך ממשק המשתמש, לערוך אותו ולקבל את קובץ הפלט הסופי.

העורך יכול לבצע את כל הפעולות של הצופה, ובנוסף אליהן:

* יצירת מפרט חדש.
* עריכת מפרט קיים.

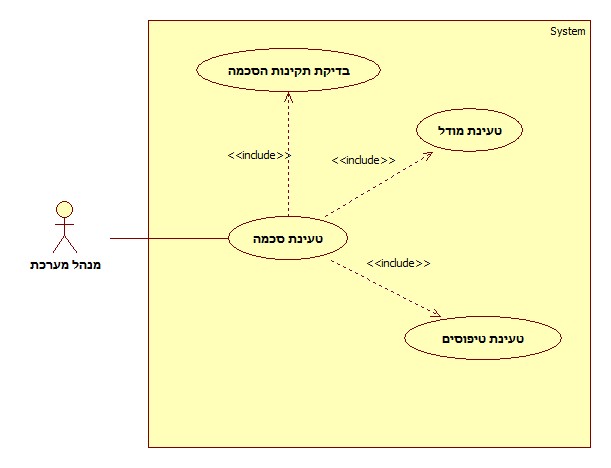
תרשים UML כללי – לתיאור הפעולות במערכת:

****

**פעולות במערכת:**

**טעינת סכמה:**

מנהל המערכת מעוניין להביא סכמה המתאימה לפרוטוקול תקשורת חדש או מעודכן. הסכמה היא קובץ XSD תקין ולא ריק, המגדיר את הטיפוסים שבהם כותב המפרט (העורך) יוכל להשתמש ולבנות את עץ ההיררכיה.

המערכת תטען את הסכמה, תוודא תקינותה ותציג את כלל הטיפוסים האפשריים על פיה. אם הסכמה אינה תקינה, תוצג הודעת שגיאה. 

**יצירת מפרט חדש:**

המשתמש רוצה לכתוב מפרט תקשורת חדש על פי סכמה מוגדרת מראש.

עליו להתחיל ליצור טיפוס אחר טיפוס, לסדר אותם בצורה היררכית ולקשר ביניהם במידת הצורך. תהליך זה ניתן לפרק למספר תתי תהליכים:

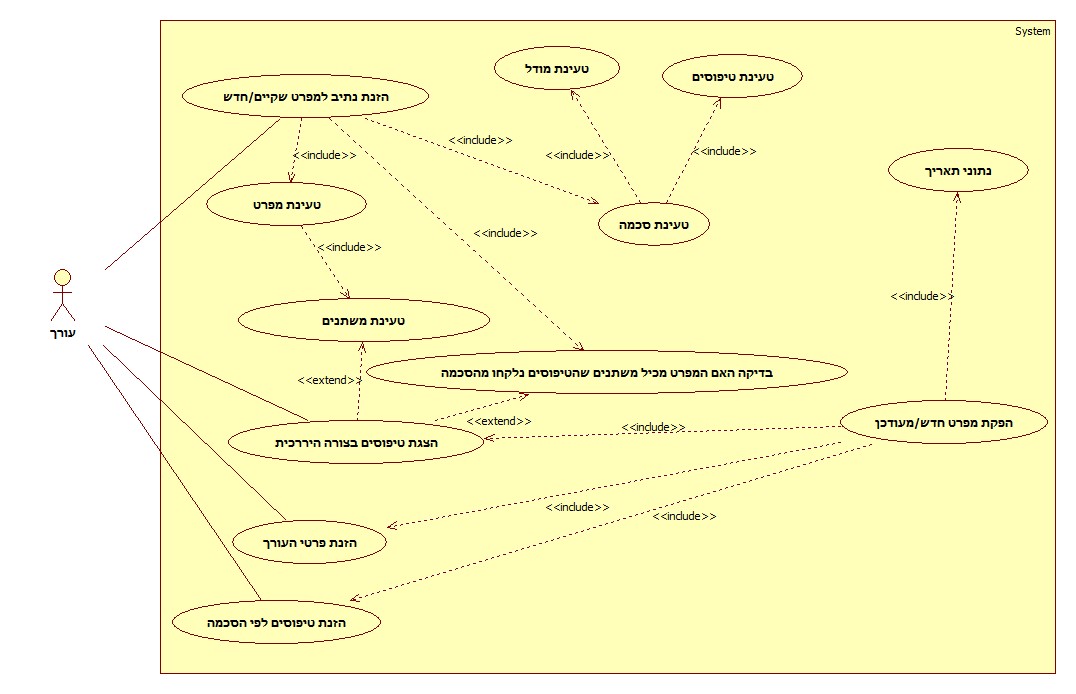
1. יצירת טיפוס פשוט, כדוגמת uint, enum, boolean ועוד.

אלה טיפוסים שהגדרתם מסתכמת במספר שורות בודדות בקובץ הפלט ולא משתמשים בטיפוסים חיצוניים. יש להם attributes הכרחיים ואופציונאליים.

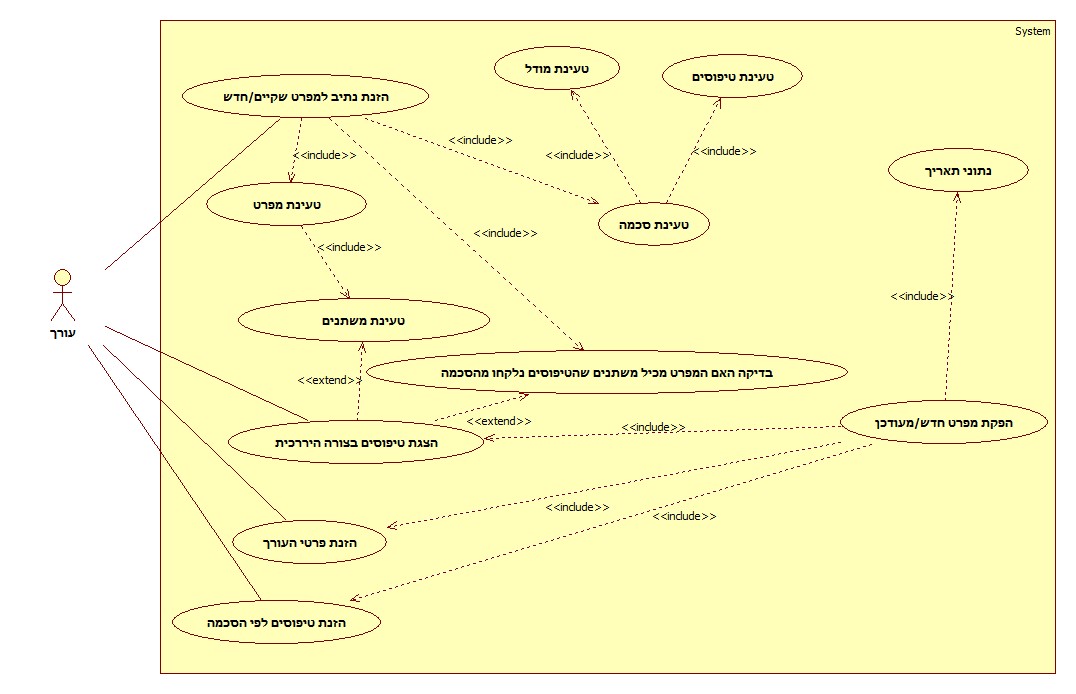
1. יצירת טיפוס בעל מבנה פנימי, כדוגמת record, array, choice. אלו הם טיפוסים מורכבים, המשתמשים בטיפוסים פשוטים ויכולים להכיל היררכיה (עץ) פנימית. גם להם attributes המגדירים אותם, למשל length עבור array. בנוסף, ישנה חוקיות בהיררכיה עבור טיפוסים מסוימים, למשל בתוך הטיפוס record יבוא הצומת member. ביצירת ההיררכיה, ניתן לבחור בטיפוסים שכבר מוגדרים על מנת לעשות reference אליהם, או ליצור טיפוס חדש בזמן. למשל, כאשר משתמש רוצה להוסיף member בתוך ה record מטיפוס enum כלשהו, הוא יכול למצוא מתוך רשימה את אותו enum מתאים או לבחור ביצירת enum חדש שיתווסף לרשימת הטיפוסים האפשריים לאחר מכן.
2. הוספת Meta Data שאינו חלק מהגדרת הסכמה. ה Meta Data הדרוש הינו קישור בין שני טיפוסים, התוכנה תאפשר בחירה מתוך רשימה של הטיפוסים הקיימים או ליצור טיפוס חדש פשוט/מורכב. בנוסף, על המשתמש להגדיר שם מזהה וסוג הקישור (מפורט בנספח), כך שמתוך ההגדרה ניתן יהיה להסיק את השדות המשותפים בין הטיפוסים. הקישור הוא לוגי, כלומר אין כל אכיפה של נכונות ה Meta Data ע"י הסכמה, אלא הוא נעשה ע"י המערכת עצמה.
3. מחיקת טיפוס קיים. אם הטיפוס מקושר לטיפוסים אחרים, באם זה ע"י reference או ע"י Meta Data, המערכת תתריע על כך ותמחק את הקישור.

לאחר גמר הוספת הטיפוסים, המשתמש ילחץ על "שמור בשם" ובכך המערכת תיצור את ה XML המבוקש ותציג אותו.

**יצירת מפרט חדש (UML):**

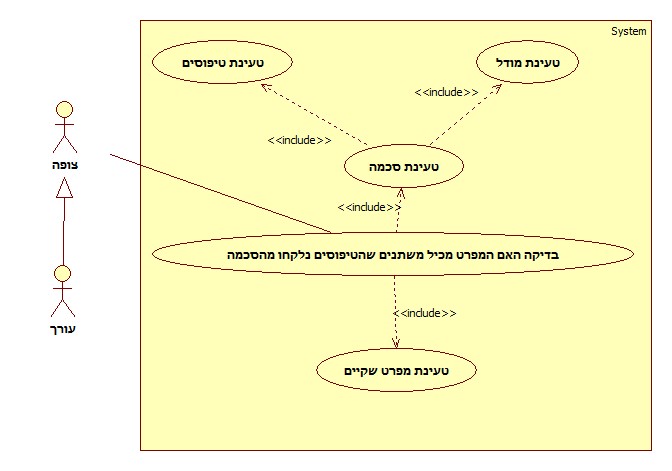


**עריכת מפרט קיים:**

המשתמש פותח מפרט XML קיים, התואם את הסכמה המוגדרת במערכת. לאחר בחירת נתיב הקובץ, המערכת תבצע ולידציה של ה XML לסכמה ותציג את היררכיית הטיפוסים שמוגדרים בו. מכאן הפעולות האפשריות זהות לאלו המתוארות עבור יצירת מפרט חדש. 

**ולידציה:**

המשתמש טוען קובץ XML ורוצה לדעת האם הקובץ תקין. המערכת תציג הודעה מתאימה: אם הקובץ תקין היא תודיע על כך, אם לא תקין – גם תפרט את מקום השגיאה.

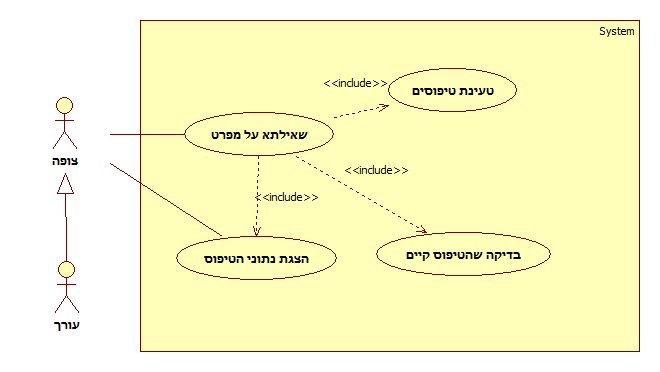


**חיפוש טיפוסים מסוג מסוים (שאילתה):**

המשתמש רוצה להכניס קריטריונים לחיפוש טיפוסים על פי מאפייניהם. ניתן לעשות את חיתוך הנתונים על פי המאפיינים הבאים (רשימה חלקית):

* סוג טיפוס.
* שם טיפוס.
* טיפוסים המשתמשים בטיפוס מסוים.
* Meta Data.
* מיקום בהיררכיה.

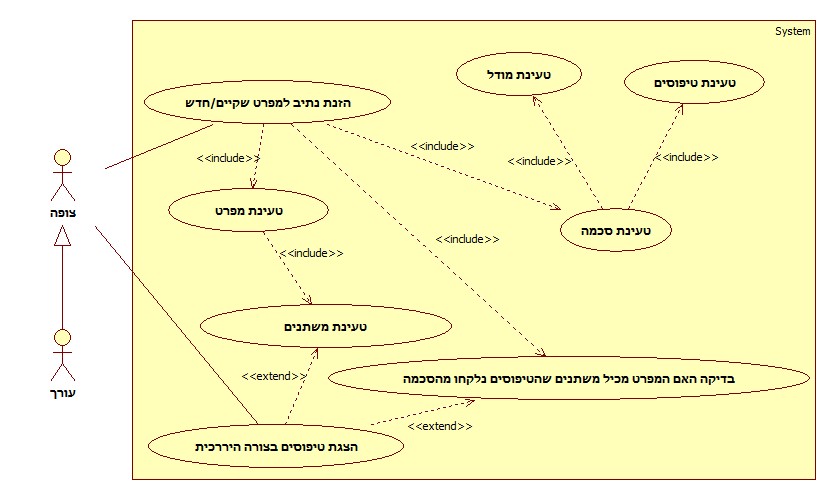
לצופה (או לעורך) מוצג מסך עם הטיפוסים העונים להגדרה המבוקשת וניתנת אפשרות לעבור למיקומו בתוך עץ ההיררכיה.



**הצגת קובץ הפלט:**

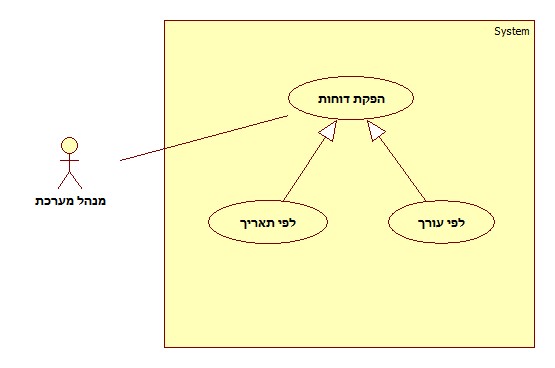
העורך או הצופה רוצה לראות את קובץ ה XML שנבנה באמצעות המערכת לאחר הבנייה או במהלכה.

המערכת טוענת את המפרט, מוודאת תקינות אל מול הסכמה ופותחת מסך עם ה XML כפי על פי בנייתו של העורך במהלך יצירת המפרט.



**הצגת דו"ח פעילות:**

מנהל המערכת רוצה לראות מה הפעילות שהתבצעה במערכת, כלומר אילו מפרטים נערכו, אילו מפרטים עודכנו וע"י מי. המערכת תאפשר ייצור דו"ח פעילות שהצגתו תכיל את שם המפרט שנערך, הסכמה שבה השתמש, המשתמש שערך והתאריך. המנהל יוכל לסדר את המידע על פי כל אחד מהמאפיינים וכך לעקוב אחרי השינויים השונים.

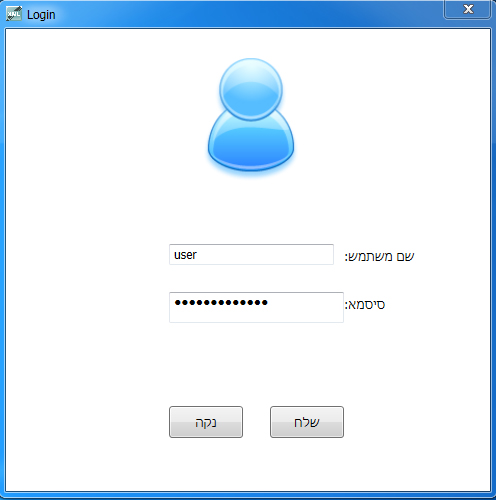


**מדריך למשתמש**

**(בליווי מסכים לדוגמה)**

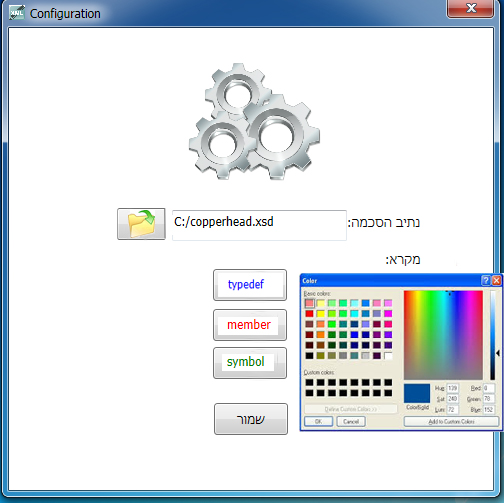
מסך כניסה

בעת הכניסה למערכת תתבקש לבחור את סוג המשתמש שלך ולהקליד את הסיסמה המתאימה. סוגי המשתמשים הם צופה, עורך ומנהל מערכת אשר ישמרו בקובץ קונפיגורציה.



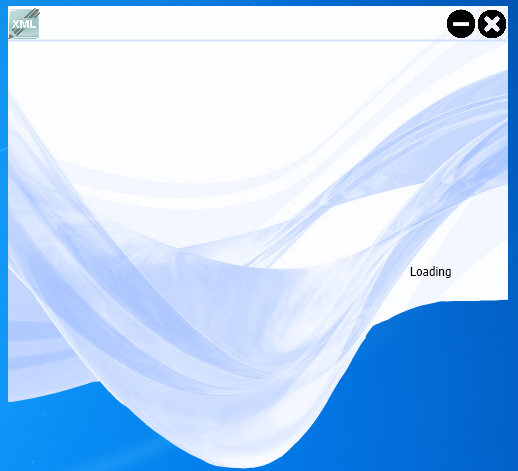
מסך קונפיגורציה

עבור מנהל המערכת תתאפשר החלפת הסכמה. ניתן לעשות זאת דרך מסך קונפיגורציה. עבור שאר המשתמשים ניתן לערוך את הצבעים של הסוגים השונים של הטיפוסים.



מסך המתנה לטעינת וטיפוסים

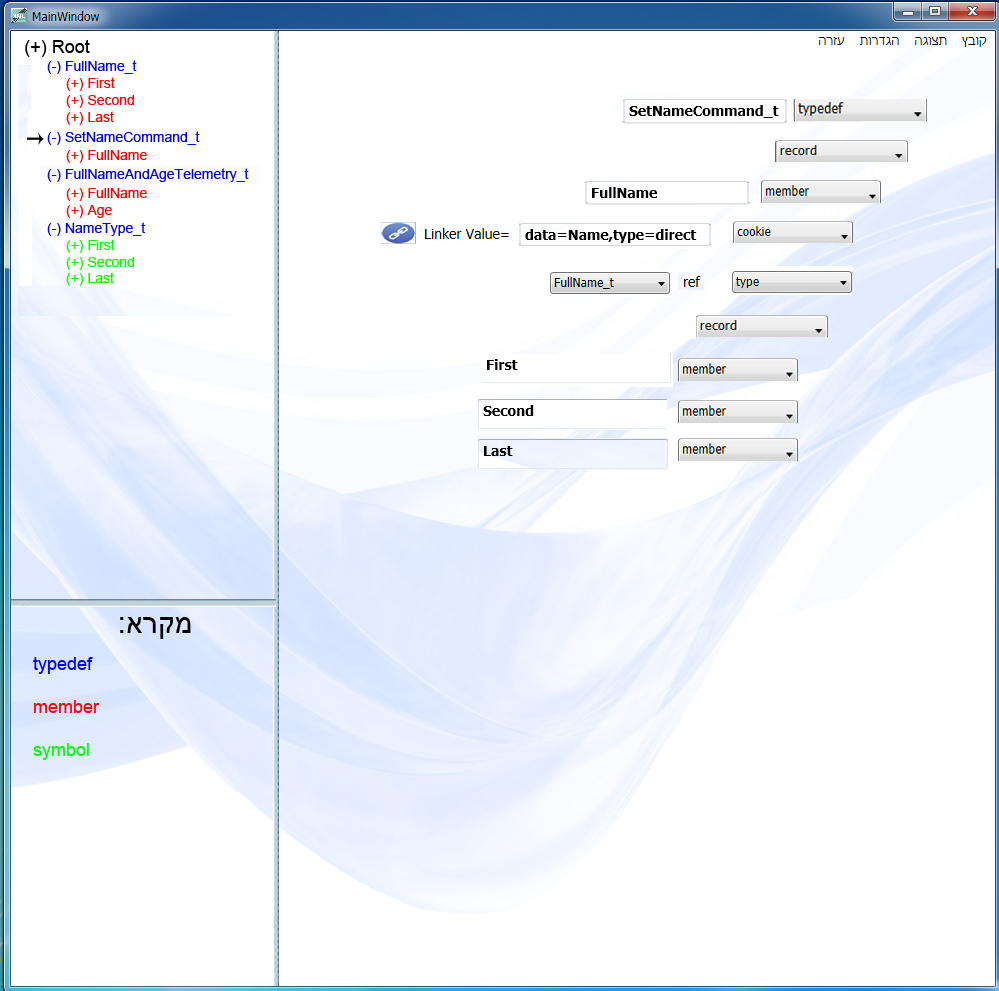
בזמן טעינת הטיפוסים השונים, שיכולים להיות מאוד מורכבים ובעלי היררכיה ורקורסיה עמוקה, יוצג מסך המתנה, שכן הפעולה עשויה לקחת זמן עיבוד רב.



מסך ראשי

זהו המסך שדרכו רוב העבודה ויכולות המערכת נעשות. המסך מחולק ל 3 אזורים:

* עץ היררכיה המייצג את עץ הטיפוסים. העץ מראה את כלל העבודה שנעשתה עד כה, כמין מפה של התוצר הסופי, שהוא המפרט.
* מקרא צבעים עבור כל מופע של טיפוס בתוך עץ ההיררכיה.
* שדה עריכה ופרטים. זהו רוב המסך, שם רואים את הטיפוסים השונים שבעץ, את מאפייניהם (attributes) ומאפשר עריכתם (עבור משתמש עורך).



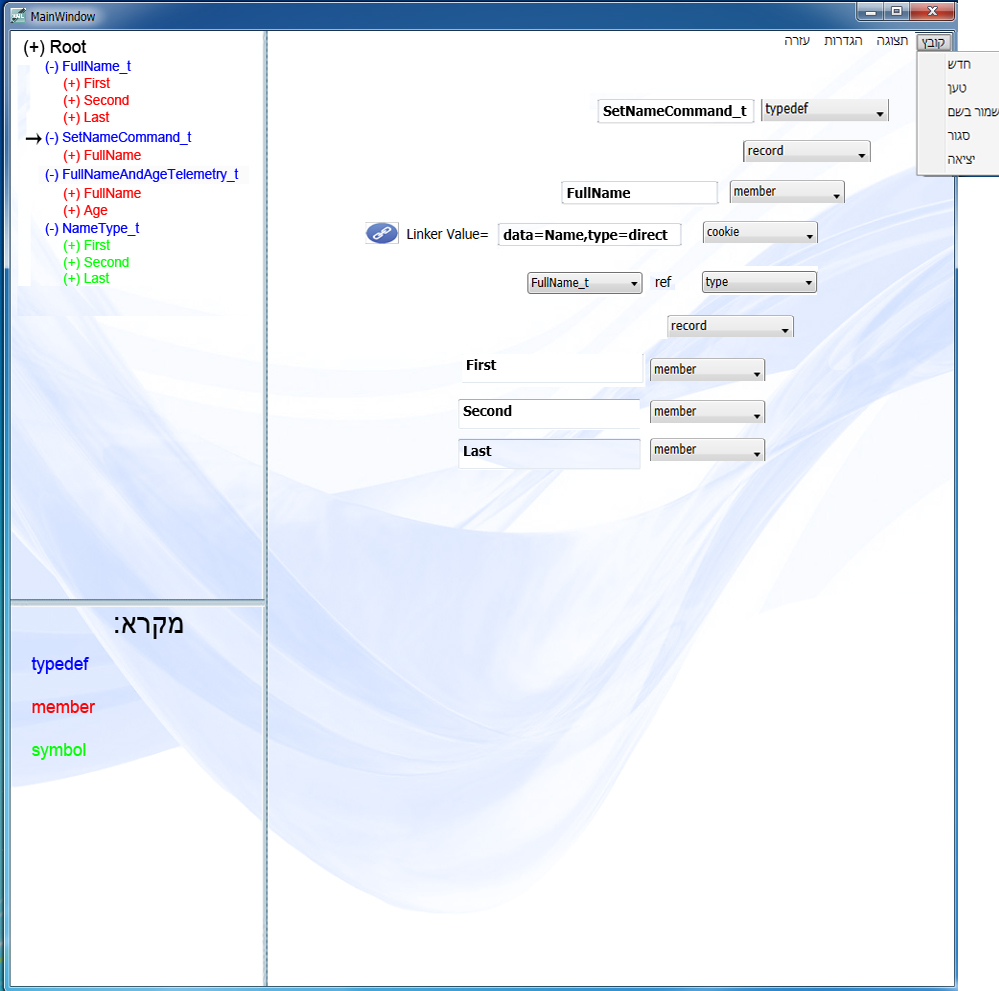
מסך ראשי – המשך

במסך הראשי ישנם מצבים שונים, הוא דינמי. בהתאם לטיפוסים האפשריים מהסכמה (XSD) ובהתאם לצורה שבה המשתמש מסדר אותם מופעים, המסך מגיב.

כמו כן, ישנו תפריט ראשי עם הפעולות שהמערכת חושפת.

עבור תפריט קובץ:

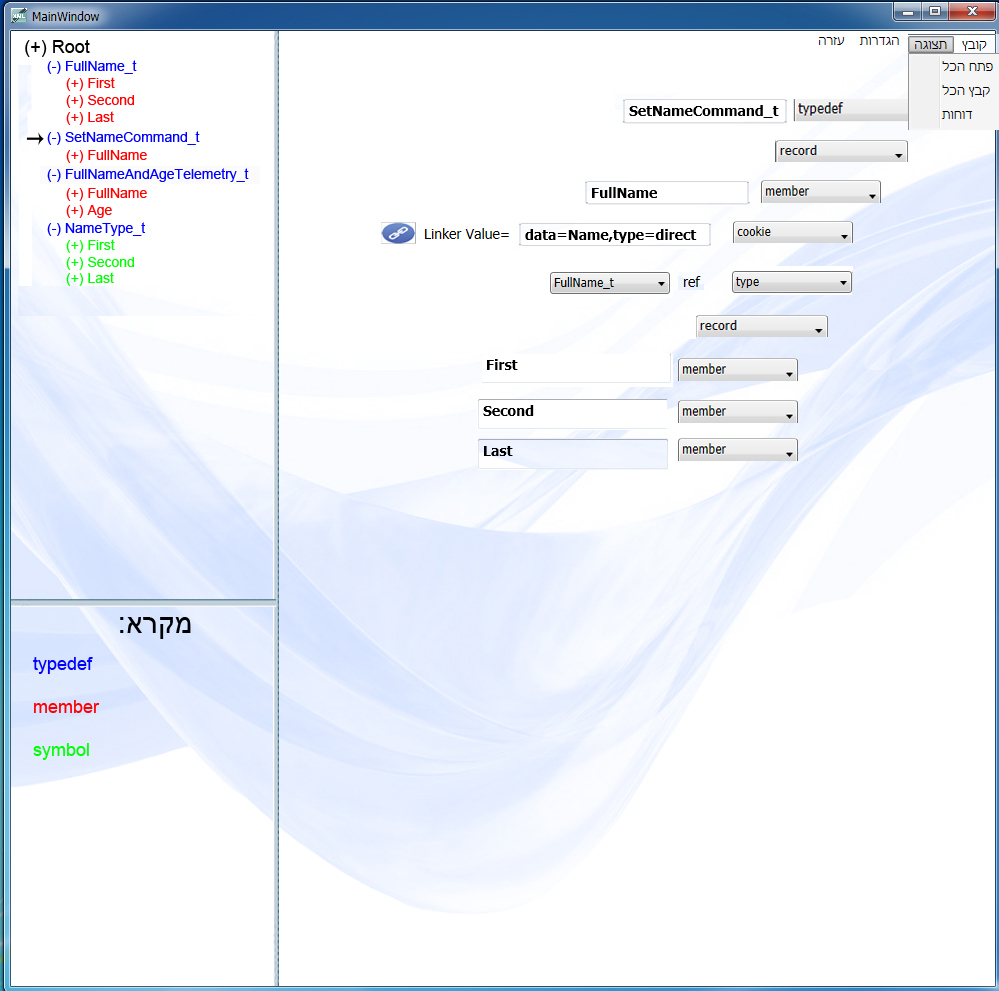
* *חדש:* יצירת מפרט XML חדש בהתאם לסכמה הטעונה.
* *טען:* לטעון קובץ XML קיים על מנת לצפות בו או לערוך אותו.
* *שמור בשם:* שמירת העבודה הנוכחית.
* *סגור:* סגירת הקובץ הנוכחי.
* *יציאה:* יציאה מהתוכנה.



מסך ראשי – המשך

תפריט תצוגה

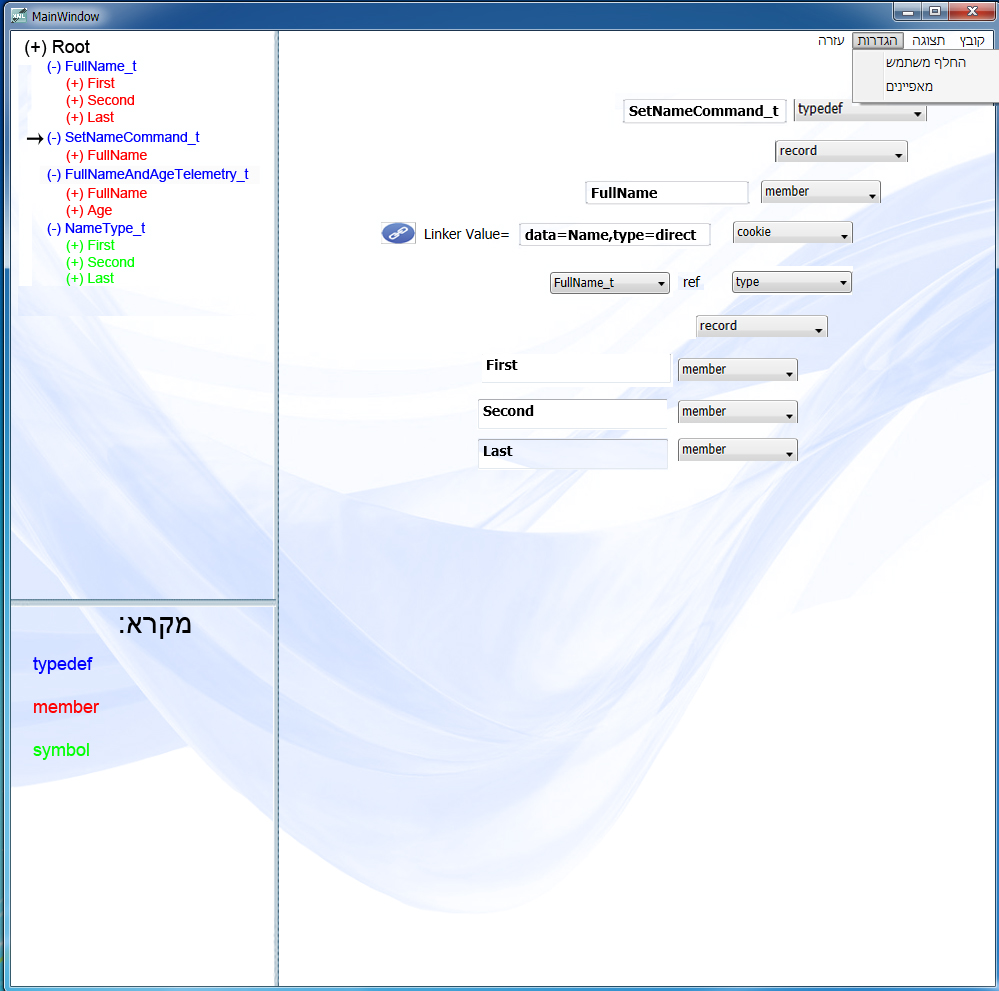
* *פתח הכל:* פותח את כל הצמתים בעץ המפרט.
* *קבץ הכל:* מקבץ את כל הצמתים בעץ המפרט.
* *דוחות*: פתיחת מסך דוחות עבור מנהל מערכת*.*



מסך ראשי – המשך

תפריט הגדרות

* *החלף משתמש:* יציאה ממשתמש אחד וחזרה למסך התחברות.
* *מאפיינים:* כניסה למסך הגדרות.



מסך ראשי – המשך

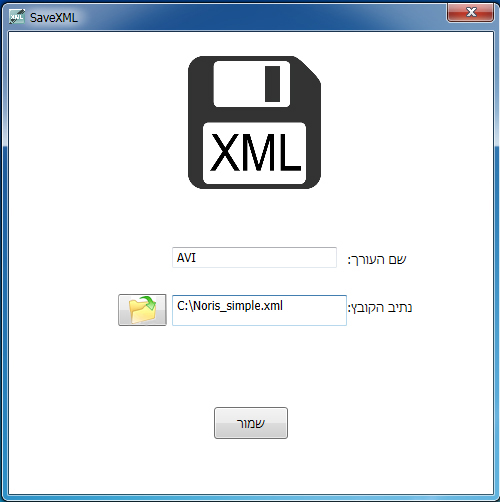
תפריט עזרה

* *אודות:* פתיחת מסך אודות.



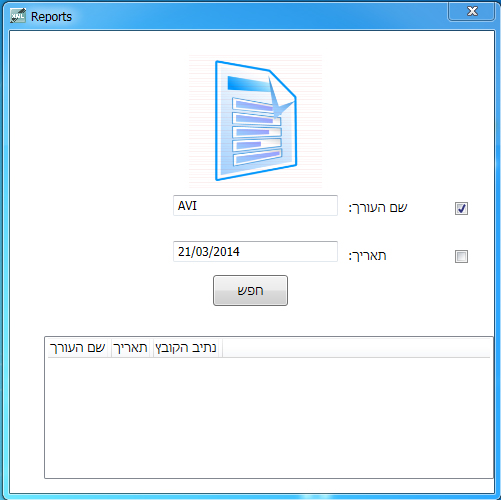
מסך שמירה

לאחר שהעורך סיים לעדכן או ליצור את המפרט, עליו לציין את שמו (עבור תיעוד ודו"חות) ואת הנתיב שהוא רוצה לשמור את הקובץ אליו.



מסך דו"חות

מנהל המערכת יכול להיכנס למערכת על מנת לבדוק את הפעילות בה. לכן באפשרותו לראות את כלל העדכונים או היצירות מפרטים על פי קריטריון לבחירתו: על פי שם עורך או על פי תאריך. לאחר בחירתו, הוא לוחץ על "חפש" ומוצגות לו רשומות העונות על הגדרתו, כולל נתיב הקובץ ששמרו אליו.



**נספחים**

**נספח א': קובץ XSD**

<!-- XML Copperhead Schema -->

<schema xmlns:ch="http://pkwiki/2005/copperhead" xmlns="http://www.w3.org/2001/XMLSchema" xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http://pkwiki/2005/copperhead" elementFormDefault="unqualified" attributeFormDefault="unqualified">

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ General Types \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<!-- uriType: define URI's as specified in the RFCs 2396 and 2732 -->

<simpleType name="uriType">

<restriction base="xsd:anyURI"/>

</simpleType>

<!-- identifierType: Variable type. In addition can not start with "\_" or contain "\_\_" -->

<simpleType name="identifierType">

<restriction base="xsd:string">

<pattern value="[A-Za-z]\_?([A-Za-z0-9]+\_\*)\*"/>

</restriction>

</simpleType>

<!-- integerLiteralType: Numeric integer type + Hexadecimal type -->

<simpleType name="integerLiteralType">

<restriction base="xsd:string">

<pattern value="-?([0-9]+|0[xX][0-9A-Fa-f]+)"/>

</restriction>

</simpleType>

<!-- versionType: digit.digit -->

<simpleType name="versionType">

<restriction base="xsd:string">

<pattern value="[0-9]+(\.[0-9]+)\*"/>

</restriction>

</simpleType>

<!-- tempQualifiedIdentifierType: Identifier.Identifier -->

<simpleType name="tempQualifiedIdentifierType">

<restriction base="xsd:string">

<pattern value="([A-Za-z]\_?([A-Za-z0-9]+\_?)\*)\.([A-Za-z]\_?([A-Za-z0-9]+\_?)\*)"/>

</restriction>

</simpleType>

<!-- qualifiedIdentifierType: Identifier or Identifier.Identifier -->

<simpleType name="qualifiedIdentifierType">

<union memberTypes="ch:identifierType ch:tempQualifiedIdentifierType"/>

</simpleType>

<!-- fullPathIdentifierType: Identifier;Identifier|Identifier\..\Identifier -->

<simpleType name="fullPathIdentifierType">

<restriction base="xsd:string">

<pattern value="([A-Za-z]\_?([A-Za-z0-9]+\_\*)\*[;|\\])\*([A-Za-z]\_?([A-Za-z0-9]+\_\*)\*)"/>

</restriction>

</simpleType>

<!-- expressionType: integerLiteralType or qualifiedIdentifierType -->

<!-- uses: for elements or attributes which get value and should be defined as -->

<!-- Numeric or Hexadecimal types or qualifiedIdentifierType for using Consts. -->

<simpleType name="expressionType">

<union memberTypes="ch:integerLiteralType ch:qualifiedIdentifierType"/>

</simpleType>

<simpleType name="constExpressionType">

<restriction base="xsd:string">

<pattern value="-?(\()\*([0-9]+|0[xX][0-9A-Fa-f]+|[A-Za-z\_0-9]+(.[A-Za-z\_0-9]+)\*)(\))\*(\s\*([-+/\*]|//|\\*\\*)\s\*(\()\*([0-9]+|0[xX][0-9A-Fa-f]+|[A-Za-z\_0-9]+)(.[A-Za-z\_0-9]+)\*(\))\*)\*"/>

</restriction>

</simpleType>

<!-- alignmentType: packed or unpacked -->

<simpleType name="alignmentType">

<restriction base="xsd:string">

<enumeration value="packed"/>

<enumeration value="unpacked"/>

</restriction>

</simpleType>

<simpleType name="endiannessType">

<restriction base="xsd:string">

<enumeration value="big"/>

<enumeration value="little"/>

</restriction>

</simpleType>

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Document Type \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<!-- documentType: Documentation of an element. -->

<!-- Documentation's type is string and should be inserted under the element which -->

<!-- we want to document. -->

<simpleType name="documentType">

<restriction base="xsd:string"/>

</simpleType>

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Import Type \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<!-- CopperheadSpecification allows us to import other specifications into the module. -->

<!-- Importing is done using import element inside the module. -->

<complexType name="importType">

<attribute name="namespace" type="ch:identifierType" use="required"/>

<attribute name="href" type="ch:uriType" use="required"/>

</complexType>

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Based Types \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<complexType name="Container"/>

<complexType name="Repeating"/>

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ TypeRef Type \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<!-- This is a reference to a predefined typedef. It is used associate a common -->

<!-- typedef into several locations in the module's -->

<complexType name="typeRefType">

<attribute name="ref" type="ch:qualifiedIdentifierType" use="required"/>

</complexType>

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Record Types \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<!-- Record is a list of fields, called members, grouped together into a single -->

<!-- logical group. -->

<!-- Member defines a field to the record and attach a name to it. Members can contain -->

<!-- any type so nested records are allowed. -->

<complexType name="memberType">

<sequence>

<group ref="ch:allTypes"/>

</sequence>

<attribute name="name" type="ch:identifierType" use="required"/>

<attribute name="offset" type="ch:expressionType" use="optional"/>

</complexType>

<!-- Pad defines specific number of bits as a constant value -->

<complexType name="padType">

<attribute name="bits" type="ch:expressionType" use="required"/>

<attribute name="value" type="ch:expressionType" use="optional"/>

<attribute name="endianness" type="ch:endiannessType" use="optional"/>

</complexType>

<!-- Skip is used when those bits are really don't care's. -->

<complexType name="skipType">

<attribute name="bits" type="ch:expressionType" use="required"/>

</complexType>

<!-- Contains one of the elements: member, pad, skip. -->

<group name="recordElement">

<choice>

<element name="member" type="ch:memberType"/>

<element name="pad" type="ch:padType"/>

<element name="skip" type="ch:skipType"/>

</choice>

</group>

<!-- Record is a list of fields, called members, grouped together into a single -->

<!-- logical group. -->

<complexType name="recordType">

<complexContent>

<extension base="ch:Container">

<sequence>

<element name="doc" type="ch:documentType" minOccurs="0" maxOccurs="unbounded"/>

<element name="cookie" type="ch:cookieType" minOccurs="0" maxOccurs="unbounded"/>

<group ref="ch:recordElement" minOccurs="0" maxOccurs="unbounded"/>

</sequence>

<attribute name="extends" type="ch:qualifiedIdentifierType" use="optional"/>

<attribute name="align" type="ch:alignmentType" use="optional"/>

</extension>

</complexContent>

</complexType>

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Choice Types \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<!-- Variant is a single option for a choice. -->

<complexType name="variantElement">

<sequence>

<element name="doc" type="ch:documentType" minOccurs="0" maxOccurs="unbounded"/>

<element name="cookie" type="ch:cookieType" minOccurs="0" maxOccurs="unbounded"/>

<group ref="ch:allDataTypes" minOccurs="0"/>

</sequence>

<attribute name="name" type="ch:identifierType" use="required"/>

<attribute name="tag\_bits" type="ch:constExpressionType" use="optional"/>

<attribute name="tag" type="ch:constExpressionType" use="optional"/>

<attribute name="response" type="ch:fullPathIdentifierType" use="optional"/>

</complexType>

<!-- Choice defines a branch in the message structure. Choice contain a tag which -->

<!-- definens to which if the options, called variants, the choice refers. -->

<complexType name="choiceType">

<complexContent>

<extension base="ch:Container">

<sequence>

<element name="doc" type="ch:documentType" minOccurs="0" maxOccurs="unbounded"/>

<element name="cookie" type="ch:cookieType" minOccurs="0" maxOccurs="unbounded"/>

<element name="variant" type="ch:variantElement" minOccurs="0" maxOccurs="unbounded"/>

</sequence>

<attribute name="tag\_bits" type="ch:expressionType" use="optional"/>

<attribute name="align" type="ch:alignmentType" use="optional"/>

<attribute name="endianness" type="ch:endiannessType" use="optional"/>

</extension>

</complexContent>

</complexType>

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Primitive Types \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<!-- base primitive type -->

<complexType name="primitiveType" abstract="true"/>

<!-- ordinalType: defines integer, boolean and char types. -->

<complexType name="ordinalType" abstract="true">

<complexContent>

<extension base="ch:primitiveType">

<attribute name="bits" type="ch:expressionType" use="optional"/>

<attribute name="endianness" type="ch:endiannessType" use="optional"/>

</extension>

</complexContent>

</complexType>

<!-- integerType: defines general integer type. -->

<complexType name="integerType" abstract="true">

<complexContent>

<extension base="ch:ordinalType">

<attribute name="max" type="ch:constExpressionType" use="optional"/>

<attribute name="min" type="ch:constExpressionType" use="optional"/>

</extension>

</complexContent>

</complexType>

<!-- integerType: defines specific integer type - unsigned int. -->

<complexType name="uintType">

<complexContent>

<extension base="ch:integerType"/>

</complexContent>

</complexType>

<!-- boolType: defines boolean type - 'true' or 'false' value -->

<complexType name="boolType">

<complexContent>

<extension base="ch:primitiveType">

<attribute name="bits" type="ch:expressionType" use="optional"/>

</extension>

</complexContent>

</complexType>

<!-- charType: unsigned number,spanned over 8 bits, that defines an ASCII character -->

<complexType name="charType">

<complexContent>

<restriction base="ch:primitiveType"/>

</complexContent>

</complexType>

<!-- Zero terminated string of ASCII characters, like C char\*. Every character is -->

<!-- encoded in 8 bits, including the zero at the end. -->

<complexType name="stringType">

<complexContent>

<extension base="ch:primitiveType">

<attribute name="max\_length" type="ch:expressionType" use="required"/>

</extension>

</complexContent>

</complexType>

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Sequence Type \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<!-- Sequence is basically an Array with variable length. Sequence is encoded as -->

<!-- it's length as the first n bits, followed by a stream of all it's items. -->

<complexType name="sequenceType">

<complexContent>

<extension base="ch:Repeating">

<sequence>

<group ref="ch:allTypes"/>

</sequence>

<attribute name="max\_length" type="ch:expressionType" use="optional"/>

<attribute name="length\_bits" type="ch:expressionType" use="optional"/>

<attribute name="align" type="ch:alignmentType" use="optional"/>

<attribute name="omit\_length" type="xsd:boolean" use="optional"/>

<attribute name="endianness" type="ch:endiannessType" use="optional"/>

</extension>

</complexContent>

</complexType>

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Enum Types \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<!-- A list of (symbolic value, numerical value) pairs. -->

<complexType name="enumType">

<sequence>

<element name="symbol" type="ch:symbolType" minOccurs="0" maxOccurs="unbounded"/>

</sequence>

<attribute name="bits" type="ch:expressionType" use="optional"/>

<attribute name="endianness" type="ch:endiannessType" use="optional"/>

</complexType>

<!-- Each symbol represents a (logical value, numerical value) pair in the enum. -->

<complexType name="symbolType">

<sequence>

<element name="doc" type="ch:documentType" minOccurs="0" maxOccurs="unbounded"/>

</sequence>

<attribute name="name" type="ch:identifierType" use="required"/>

<attribute name="tag" type="ch:expressionType" use="optional"/>

</complexType>

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Cookie Type \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<complexType name="cookieType">

<sequence>

<element name="doc" type="ch:documentType" minOccurs="0" maxOccurs="unbounded"/>

</sequence>

<attribute name="name" type="ch:identifierType" use="required"/>

<attribute name="value" type="xsd:string" use="optional"/>

</complexType>

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_All Types \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<!-- allTypes: group which contains all types which useful for some complex types -->

<!-- as sequence, array, etc. Plus the optional doc and cookie elements. -->

<group name="allTypes">

<sequence>

<choice minOccurs="0" maxOccurs="unbounded">

<element name="doc" type="ch:documentType" minOccurs="0" maxOccurs="unbounded"/>

<element name="cookie" type="ch:cookieType" minOccurs="0" maxOccurs="unbounded"/>

</choice>

<group ref="ch:allDataTypes"/>

</sequence>

</group>

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_All Types \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<!-- allDataTypes: group which contains all types which useful for some complex types -->

<!-- as sequence, array, etc. -->

<group name="allDataTypes">

<sequence>

<choice>

<element name="type" type="ch:typeRefType"/>

<element name="record" type="ch:recordType"/>

<element name="choice" type="ch:choiceType"/>

<element name="sequence" type="ch:sequenceType"/>

<element name="enum" type="ch:enumType"/>

<element name="uint" type="ch:uintType"/>

<element name="bool" type="ch:boolType"/>

<element name="char" type="ch:charType"/>

<element name="string" type="ch:stringType"/>

<element name="pad" type="ch:padType"/>

<element name="skip" type="ch:skipType"/>

</choice>

</sequence>

</group>

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Typedef Type \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<!-- Types define how binary raw data is spanned across one or more logical fields.-->

<!-- Using TypeDef, one can assign a name to a type so it can be used later on in -->

<!-- more then one occasion. -->

<complexType name="typedefType">

<sequence>

<group ref="ch:allTypes"/>

</sequence>

<attribute name="name" type="ch:identifierType" use="required"/>

</complexType>

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Module Type \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<!-- Any CopperheadSpecification file must srart with a root element called module. -->

<!-- The module element is designed to gather all the module into a single element, -->

<!-- with a given name and version. -->

<element name="module">

<complexType>

<sequence>

<choice minOccurs="0" maxOccurs="unbounded">

<element name="doc" type="ch:documentType" minOccurs="0" maxOccurs="unbounded"/>

<element name="import" type="ch:importType" minOccurs="0" maxOccurs="unbounded"/>

<element name="cookie" type="ch:cookieType" minOccurs="0" maxOccurs="unbounded"/>

</choice>

<choice minOccurs="0" maxOccurs="unbounded"> <!-- without choice the order should be implied and typedef cannot be after compdef for example -->

<element name="typedef" type="ch:typedefType" minOccurs="0" maxOccurs="unbounded"/>

</choice>

</sequence>

<attribute name="name" type="ch:identifierType" use="required"/>

<attribute name="version" type="ch:versionType" use="optional"/>

<attribute name="endianness" type="ch:endiannessType" use="optional"/>

<attribute name="inherits" type="ch:identifierType" use="optional"/>

</complexType>

</element>

</schema>

**נספח ב': מפרט XML מבוסס XSD**

<ch:module name="Norris">

<typedef name="FullName\_t">

<record>

<member name="First">

<type ref="NameEntity\_t"/>

</member>

<member name="Second">

<type ref="NameEntity\_t"/>

</member>

<member name="Last">

<type ref="NameEntity\_t"/>

</member>

</record>

</typedef>

<typedef name="NameEntity\_t">

<string max\_length="20" />

</typedef>

<typedef name="NameType\_t">

<enum bits="8">

<symbol name="First" tag="0" />

<symbol name="Second" tag="1" />

<symbol name="Last" tag="2" />

</enum>

</typedef>

<typedef name= "SetNameCommand\_t">

<record>

<member name="FullName">

<cookie name="linker" value="data=Name,type=direct"/>

<type ref="FullName\_t" />

</member>

</record>

</typedef>

<typedef name= "FulltNameAndAgeTelememetry\_t">

<record>

<member name="FullName">

<cookie name="linker" value="data=Name,type=direct"/>

<type ref="FullName\_t" />

</member>

<member name="Age">

<unit bits="16"/>

</member>

</record>

</typedef>

</ch:module>