3GPP TSG-T (Terminals) Meeting #11
Palm Springs, USA, 14 - 16 March, 2001

Tdoc TP-010070

Source: T3

Title: Change Requests to GSM 03.19 "SIM API for Java Card™"

Document for: Approval

This document contains change requests to GSM 03.19 v7.4.0 and v8.0.0 agreed by T3.

T3 Doc	Spec	CR	Rv	Rel	Subject
T3-010251	03.19	A011		R98	Clarification about ArrayIndexOutOfBoundsException
T3-010252	03.19	A012		R99	Clarification about ArrayIndexOutOfBoundsException

3GPP TSG-T3 Meeting #18
Sofia Antipolis, France, 1st - 2nd of March 2001

Tdoc T3-010251

CR-Form-v3

CHANGE REQUEST

⌘	**03.19**	**CR**	**011**	⌘ rev	**-**	⌘ Current version:	7.4.0 ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM [✓ X] ME/UE ☐ Radio Access Network ☐ Core Network ☐

Title: ⌘	Clarification about ArrayIndexOutOfBoundsException		
Source: ⌘	T3		
Work item code: ⌘		***Date:*** ⌘	02/03/2001
Category: ⌘	**F**	***Release:*** ⌘	R98
	Use one of the following categories: ***F*** *(essential correction)* ***A*** *(corresponds to a correction in an earlier release)* ***B*** *(Addition of feature),* ***C*** *(Functional modification of feature)* ***D*** *(Editorial modification)* Detailed explanations of the above categories can be found in 3GPP TR 21.900.		*Use one of the following releases:* *2 (GSM Phase 2)* *R96 (Release 1996)* *R97 (Release 1997)* *R98 (Release 1998)* *R99 (Release 1999)* *REL-4 (Release 4)* *REL-5 (Release 5)*

Reason for change: ⌘	Clarify the reason for an ArrayIndexOutOfBoundException
Summary of change: ⌘	For all methods throwing an ArrayIndexOutOfBoundException introduction of a Note that explains in which case this exception is thrown
Consequences if not approved: ⌘	Can cause interoperability problems with different implementations

Clauses affected: ⌘	Annex_A_java	
Other specs affected: ⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘
Other comments: ⌘		

How to create CRs using this form:

Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://www.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2000-09 contains the specifications resulting from the September 2000 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

Package sim.access

Class SIMView

```
    /**
     * SELECT command as defined in GSM 11.11 standard.<br>
     * By default, the MF is selected at the beginning of each applet activation
     * (current file = MF). This method selects a file of the GSM file system.
     * The file search starts at the current DF of the applet according to
     * the file search method described in GSM 11.11 specification. The current
     * DF or current EF and the current record pointer of the calling applet will
     * be changed after successful execution.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>fciOffset</code><em> or </em><code>fciLength</code><em> parameter is
negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no read is performed.</em>
     * <li><em>If </em><code>fciOffset+fciLength</code><em> is greater than
</em><code>fci.length</code><em>, the length
     * of the </em><code>fci</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no status request is performed.</em>
     * </ul>
     *
     * @param fid is the File Identifier of the file to be selected.
     * @param fci is the reference to the target byte array for FCI (File Control
     *        Information) of current file, coding is according to GSM 11.11.
     *        If <code>fci</code> is <code>null</code> the <code>NullPointerException</code> is
thrown.
     * @param fciOffset is the offset in the <code>fci</code> buffer for the response data.
     * @param fciLength is the length of the required data in the <code>fci</code> byte array.
     *        If the <code>fciLength</code> is greater than the length of the response, the whole
     *        response is copied into the <code>fci</code> buffer and the length of the response
     *            is returned by the method. If the <code>fcilength</code> is smaller than the
length
     *        of the response, the first part of the response is copied into the <code>fci</code>
     *            buffer and the <code>fciLength</code> is returned by the method.
     *
     * @return length of the data which have been written in the <code>fci</code> buffer
     *         (cannot be greater than <code>fciLength</code> parameter)
     *
     * @exception NullPointerException if <code>fci</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException if operation would cause access of data outside
     *         array bounds.
     * @exception SIMViewException in case of error<ul>
     *          <li><code>FILE_NOT_FOUND</code>
     *          <li><code>MEMORY_PROBLEM</code>
     *          <li><code>INTERNAL_ERROR</code></ul>
     */
    public short select(short fid,
                        byte  fci[],
                        short fciOffset,
                        short fciLength) throws NullPointerException,
                                                ArrayIndexOutOfBoundsException,
                                                SIMViewException;


    /**
     * STATUS command as defined in GSM 11.11 standard.<br>
     * This method returns the FCI (File Control Information) of the current DF
     * (or MF) of the calling applet.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>fciOffset</code><em> or </em><code>fciLength</code><em> parameter is
negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no read is performed.</em>
     * <li><em>If </em><code>fciOffset+fciLength</code><em> is greater than
</em><code>fci.length</code><em>, the length
     * of the </em><code>fci</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no status request is performed.</em>
     * </ul>
     *
     * @param fci is the reference to the target byte array for FCI (File Control
     *        Information) of current DF (or MF), coding is according to GSM 11.11.
     *        If <code>fci</code> is <code>null</code> the <code>NullPointerException</code> is
thrown.
     * @param fciOffset is the offset in the <code>fci</code> buffer for the response data.
     * @param fciLength is the length of the required data in the <code>fci</code> byte array.
         *        If the <code>fciLength</code> is greater than the length of the response, the
whole
         *        response is copied into the <code>fci</code> buffer and the length of the response
         *            is returned by the method. If the <code>fcilength</code> is smaller than the
length
         *        of the response, the first part of the response is copied into the
<code>fci</code>
         *            buffer and the <code>fciLength</code> is returned by the method.
     *
     * @return length of the data which have been written in the <code>fci</code> buffer
     *         (cannot be greater than <code>fciLength</code> parameter)
     *
     * @exception NullPointerException if <code>fci</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException if operation would cause access of data outside
         *         array bounds.
     * @exception SIMViewException in case of error <ul>
     *          <li><code>MEMORY_PROBLEM</code>
     *          <li><code>INTERNAL_ERROR</code></ul>
     */
    public short status(byte  fci[],
                        short fciOffset,
                        short fciLength) throws NullPointerException,
                                                ArrayIndexOutOfBoundsException,
                                                SIMViewException;

    /**
     * READ BINARY command as defined in GSM 11.11 standard.<br>
     * This method reads the data bytes of the current transparent EF of the
     * calling applet.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>respOffset</code><em> or </em><code>respLength</code><em> parameter is
negative a </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no read is performed.</em>
     * <li><em>If </em><code>respOffset+respLength</code><em>is greater than
</em><code>resp.length</code><em>, the length
     * of the </em><code>resp</code><em> array a
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no select is performed.</em>
     * </ul>
     *
     * @param fileOffset is the offset in the source transparent file.
     * @param resp is the reference to the response byte array for read data.
```
~~*   If <code>resp</code> is <code>null</code> the <code>NullPointerException</code> is thrown.~~
~~*   If <code>respOffset</code> or <code>respLength</code> or <code>respOffset+respLength</code> is in contradiction~~
~~*   with the <code>resp</code> byte array the <code>ArrayIndexOutOfBoundsException</code> is thrown.~~
```
     * @param respOffset is the offset in the response byte array.
     * @param respLength is the number of bytes to read.
     *
     * @return <code>respOffset+respLength</code>
     *
     * @exception NullPointerException if <code>resp</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     *           if reading would cause access of data outside array bounds
     * @exception SIMViewException in case of error <ul>
     *          <li><code>NO_EF_SELECTED</code>
     *          <li><code>FILE_INCONSISTENT</code>
     *          <li><code>AC_NOT_FULFILLED</code>
     *          <li><code>INVALIDATION_STATUS_CONTRADICTION</code>
     *          <li><code>OUT_OF_FILE_BOUNDARIES</code>
     *          <li><code>MEMORY_PROBLEM</code>
     *          <li><code>INTERNAL_ERROR</code></ul>
     */
    public short readBinary(short fileOffset,
                            byte  resp[],
                            short respOffset,
                            short respLength) throws NullPointerException,
                                                     ArrayIndexOutOfBoundsException,
                                                     SIMViewException;
    /**
     * UPDATE BINARY command as defined in GSM 11.11 standard.<br>
     * This method updates the data bytes of the current transparent EF of
     * the calling applet.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>dataOffset</code><em> or </em><code>dataLength</code><em> parameter is
negative a </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no update is performed.</em>
     * <li><em>If </em><code>dataOffset+dataLength</code><em> is greater than
</em><code>data.length</code><em>, the length
     * of the </em><code>data</code><em> array a
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no update is performed.</em>
     * </ul>
     *
     * @param fileOffset is the offset in the destination transparent file.
     * @param data is the reference to the source byte array for data to update.
~~     *        If <code>data</code> is <code>null</code> the <code>NullPointerException</code> is
thrown.~~
~~     *        If <code>dataOffset</code> or <code>dataLength</code> or
<code>dataOffset+dataLength</code> is in contradiction~~
~~     *        with the <code>data</code> byte array the <code>ArrayIndexOutOfBoundsException</code>
is thrown.~~
     * @param dataOffset is the offset in the source byte array.
     * @param dataLength is the number of bytes to update.
     *
     * @exception NullPointerException if <code>data</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     *      if updating would cause access of data outside array bounds
     * @exception SIMViewException in case of error<ul>
     *          <li><code>NO_EF_SELECTED</code>
     *          <li><code>FILE_INCONSISTENT</code>
     *          <li><code>AC_NOT_FULFILLED</code>
     *          <li><code>INVALIDATION_STATUS_CONTRADICTION</code>
     *          <li><code>OUT_OF_FILE_BOUNDARIES</code>
     *          <li><code>MEMORY_PROBLEM</code>
     *          <li><code>INTERNAL_ERROR</code></ul>
     */
    public void updateBinary(short fileOffset,
                             byte  data[],
                             short dataOffset,
                             short dataLength) throws   NullPointerException,
                                                        ArrayIndexOutOfBoundsException,
                                                        SIMViewException;

    /**
     * READ RECORD command as defined in GSM 11.11 standard.<br>
     * This method reads the data bytes of the current linear fixed/cyclic EF
     * of the calling applet. The current record pointer can be changed due to
     * the choosen mode.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>respOffset</code><em> or </em><code>respLength</code><em> parameter is
negative a </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no read is performed.</em>
     * <li><em>If </em><code>respOffset+respLength</code><em> is greater than
</em><code>resp.length</code><em>, the length
     * of the </em><code>resp</code><em> array a
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no select is performed.</em>
     * </ul>
     *
     * @param recNumber is the record number.
     * @param mode is the mode for reading record, according to GSM 11.11.
     *        If <code>mode</code> is <code>REC_ACC_MODE_NEXT</code> and the record pointer is at
the last record
         *        the <code>RECORD_NUMBER_NOT_AVAILABLE SIMViewException</code> shall be thrown.
         *        If <code>mode</code> is <code>REC_ACC_MODE_PREVIOUS</code> and the record pointer
is at the first record,
         *        the <code>RECORD_NUMBER_NOT_AVAILABLE SIMViewException</code> shall be thrown.
     * @param recOffset is the offset in the record for the data to read.
     * @param resp is the reference to the response byte array for read data.
~~     *        If <code>resp</code> is <code>null</code> the <code>NullPointerException</code> is
thrown.~~
```

```
~~     *        If <code>respOffset</code> or <code>respLength</code> or~~
~~<code>respOffset+respLength</code> is in contradiction~~
~~     *        with the <code>resp</code> byte array the <code>ArrayIndexOutOfBoundsException</code>~~
~~is thrown.~~
     * @param respOffset is the offset in the response byte array.
     * @param respLength is the number of bytes to read.
     *
     * @return <code>respOffset+respLength</code>
     *
     * @exception NullPointerException if <code>resp</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     *     if reading would cause access of data outside array bounds
     * @exception SIMViewException in case of error <ul>
     *          <li><code>NO_EF_SELECTED</code>
     *          <li><code>FILE_INCONSISTENT</code>
     *          <li><code>AC_NOT_FULFILLED</code>
     *          <li><code>INVALIDATION_STATUS_CONTRADICTION</code>
     *          <li><code>OUT_OF_RECORD_BOUNDARIES</code>
     *          <li><code>RECORD_NUMBER_NOT_AVAILABLE</code>
     *          <li><code>INVALID_MODE</code>
     *          <li><code>MEMORY_PROBLEM</code>
     *          <li><code>INTERNAL_ERROR</code></ul>
     */
    public short readRecord(short recNumber,
                            byte  mode,
                            short recOffset,
                            byte  resp[],
                            short respOffset,
                            short respLength) throws     NullPointerException,
                                                         ArrayIndexOutOfBoundsException,
                                                         SIMViewException;

    /**
     * UPDATE RECORD command as defined in GSM 11.11 standard.<br>
     * This method updates the data bytes of the current linear fixed/cyclic EF
     * of the calling applet. The current record pointer can be changed due to
     * the choosen mode.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>dataOffset</code><em> or </em><code>dataLength</code><em> parameter is
negative a </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no select is performed.</em>
     * <li><em>If </em><code>dataOffset+dataLength</code><em> is greater than
</em><code>data.length</code><em>, the length
     * of the </em><code>data</code><em> array a
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no update is performed.</em>
     * </ul>
     *
     * @param recNumber is the record number.
     * @param mode is the mode for updating record, according to GSM 11.11.
     *        If <code>mode</code> is <code>REC_ACC_MODE_NEXT</code> and the record pointer is at
the last record
         *        the <code>RECORD_NUMBER_NOT_AVAILABLE SIMViewException</code> shall be thrown.
         *        If <code>mode</code> is <code>REC_ACC_MODE_PREVIOUS</code> and the record pointer
is at the first record,
         *        the <code>RECORD_NUMBER_NOT_AVAILABLE SIMViewException</code> shall be thrown.
     * @param recOffset is the offset in the record for the data to update.
     * @param data is the reference to the source byte array for data to update.
~~     *        If <code>data</code> is <code>null</code> the <code>NullPointerException</code> is~~
~~thrown.~~
~~     *        If <code>dataOffset</code> or <code>dataLength</code> or~~
~~<code>dataOffset+dataLength</code> is in contradiction~~
~~     *        with the <code>data</code> byte array the <code>ArrayIndexOutOfBoundsException</code>~~
~~is thrown.~~
     * @param dataOffset is the offset in the source byte array.
     * @param dataLength is the number of bytes to update.
     *
     * @exception NullPointerException if <code>data</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     *      if updating would cause access of data outside array bounds
     * @exception SIMViewException in case of error<ul>
     *          <li><code>NO_EF_SELECTED</code>
     *          <li><code>FILE_INCONSISTENT</code>
     *          <li><code>AC_NOT_FULFILLED</code>
     *          <li><code>INVALIDATION_STATUS_CONTRADICTION</code>
     *          <li><code>OUT_OF_RECORD_BOUNDARIES</code>
     *          <li><code>RECORD_NUMBER_NOT_AVAILABLE</code>
     *          <li><code>INVALID_MODE</code>
     *          <li><code>MEMORY_PROBLEM</code>
     *          <li><code>INTERNAL_ERROR</code></ul>
     */
    public void updateRecord(short recNumber,
                             byte  mode,
                             short recOffset,
                             byte  data[],
                             short dataOffset,
                             short dataLength) throws   NullPointerException,
                                                        ArrayIndexOutOfBoundsException,
                                                        SIMViewException;

    /**
     * SEEK command as defined in GSM 11.11 standard.<br>
     * This method seeks a pattern in the current linear fixed EF of the calling
     * applet.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>pattOffset</code><em> or </em><code>pattLength</code><em> parameter is
negative a </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no seek is performed.</em>
     * <li><em>If </em><code>pattOffset+pattLength</code><em> is greater than
</em><code>patt.length</code><em>, the length
     * of the </em><code>patt</code><em> array a
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no seek is performed.</em>
     * </ul>
     *
     * @param mode is the seek mode, according to GSM 11.11 (no type information).
     *          If <code>mode</code> is <code>SEEK_FROM_NEXT_FORWARD</code> and the record pointer is
at the last record
        *          the <code>PATTERN_NOT_FOUND SIMViewException</code> shall be thrown.
        *          If <code>mode</code> is <code>SEEK_FROM_PREVIOUS_BACKWARD</code> and the record
pointer is at the first record,
        *          the <code>PATTERN_NOT_FOUND SIMViewException</code> shall be thrown.
     * @param patt is the reference to the byte array containing the seek pattern.
~~     *          If <code>patt</code> is <code>null</code> the <code>NullPointerException</code> is
thrown.~~
~~     *          If <code>pattOffset</code> or <code>pattLength</code> or
<code>pattOffset+pattLength</code> is in contradiction~~
~~     *          with the <code>patt</code> byte array the <code>ArrayIndexOutOfBoundsException</code>
is thrown.~~
     * @param pattOffset is the offset of the seek pattern in the byte array.
     * @param pattLength is the length of the seek pattern.
     *          If <code>pattLength</code> is greater than the current record size than
        *          the <code>OUT_OF_RECORD_BOUNDARIES SIMViewException</code> shall be thrown.
        *          If <code>pattLength</code> is <code>zero</code> than <code>PATTERN_NOT_FOUND
SIMViewException</code> shall be thrown.
     *
     * @return record number if pattern found
     *
     * @exception NullPointerException if <code>patt</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     *        if seeking would cause access of data outside array bounds
     * @exception SIMViewException in case of error<ul>
     *          <li><code>NO_EF_SELECTED</code>
     *          <li><code>PATTERN_NOT_FOUND</code>
     *          <li><code>FILE_INCONSISTENT</code>
     *          <li><code>AC_NOT_FULFILLED</code>
     *          <li><code>INVALIDATION_STATUS_CONTRADICTION</code>
     *          <li><code>INVALID_MODE</code>
     *          <li><code>OUT_OF_RECORD_BOUNDARIES</code>
     *          <li><code>MEMORY_PROBLEM</code>
     *          <li><code>INTERNAL_ERROR</code></ul>
     */
    public short seek(byte   mode,
                      byte[] patt,
                      short  pattOffset,
                      short  pattLength) throws  NullPointerException,
                                                ArrayIndexOutOfBoundsException,
                                                SIMViewException;

    /**
     * INCREASE command as defined in GSM 11.11 standard.<br>
     * This method increases the current cyclic EF record of the calling applet.
     * The response buffer will only contain the value of the increased record.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>incrOffset</code><em> or </em><code>respOffset</code><em> parameter is
negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no increase is performed.</em>
     * <li><em>If </em><code>incrOffset</code><em> is greater than
</em><code>incr.length</code><em>, the length
     * of the </em><code>incr</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no increase is performed.</em>
     * <li><em>If </em><code>respOffset</code><em> is greater than
</em><code>resp.length</code><em>, the length
     * of the </em><code>resp</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no increase is performed.</em>
     * </ul>
     *
     * @param incr is the reference to the source byte array, containing the
     *        value to add, on 3 bytes.
     * @param incrOffset is the offset in the source byte array.
     * @param resp is the reference to the response byte array for new record value.
~~*        If <code>incr</code> or <code>resp</code> is <code>null</code> the
<code>NullPointerException</code> is thrown.~~
     *        If <code>resp</code> buffer is smaller than the record size, the
<code>ArrayIndexOutOfBoundsException</code> is thrown.
     *        If <code>resp</code> buffer is bigger than the record size, the <code>resp</code>
buffer is filled
     *        with the record value and left justified
     * @param respOffset is the offset in the response byte array.
     *
     * @return length of the valid data in the <code>resp</code> buffer
     *      (cannot be greater than the record size)
     *
     * @exception NullPointerException if <code>incr</code> or <code>resp</code> is
<code>null</code>
     * @exception ArrayIndexOutOfBoundsException if increasing would cause access of data outside
array bounds
     * @exception SIMViewException in case of error<ul>
     *          <li><code>NO_EF_SELECTED</code>
     *          <li><code>FILE_INCONSISTENT</code>
     *          <li><code>AC_NOT_FULFILLED</code>
     *          <li><code>INVALIDATION_STATUS_CONTRADICTION</code>
     *          <li><code>MAX_VALUE_REACHED</code>
     *          <li><code>MEMORY_PROBLEM</code>
     *          <li><code>INTERNAL_ERROR</code></ul>
     */
    public short increase(byte[] incr,
                          short  incrOffset,
                          byte[] resp,
                          short  respOffset) throws  NullPointerException,
                                                     ArrayIndexOutOfBoundsException,
                                                     SIMViewException;
```

Package sim.toolkit

Class EditHandler

```
    /**
     * Clears the TLV list of an EditHandler and resets the current TLV selected.
     *
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy</ul>
     */
    public void clear() throws ToolkitException {
    }

    /**
     * Appends a buffer into the EditHandler buffer.
         * A successful append does not modify the TLV selected.
         * The TLV list structure of the handler should be maintained by the applet in the
```

```
             * appended array (e.g. the length of the TLV element should be coded according to ISO 7816-
6),
             * if the TLV manipulation methods are to be used afterwards with the handler.
         *
         * <p>
         * Notes:<ul>
         * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative
an </em><code>ArrayIndexOutOfBoundsException</code>
         * <em> exception is thrown and no append is performed.</em>
         * <li><em>If </em><code>offset+length</code><em> is greater than
</em><code>buffer.length</code><em>, the length
         * of the </em><code>buffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
         * and no append is performed.</em>
         * </ul>
         *
         * @param buffer the buffer containing data for copy
         * @param offset the offset in the buffer
         * @param length the value length of the buffer
         *
         * @exception NullPointerException if <code>buffer</code> is <code>null</code>
         * @exception ArrayIndexOutOfBoundsException ~~if <code>offset</code> or <code>length</code>~~

     ~~    *        or both~~*  - if append would cause access of data outside array bounds~~, or if
<code>length</code> is negative~~.
         * @exception ToolkitException with the following reason codes: <ul>
         *      <li><code>HANDLER_OVERFLOW</code> if the EditHandler buffer is to small to append the
requested data
         *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy</ul>
         */
    public void appendArray(byte[] buffer,
                            short offset,
                            short length) throws NullPointerException,
                                                 ArrayIndexOutOfBoundsException,
                                                 ToolkitException {
    }

    /**
         * Appends a TLV element to the current TLV list (byte array format).
         * A successful append does not modify the TLV selected.
         *
         * <p>
         * Notes:<ul>
         * <li><em>If </em><code>valueOffset</code><em> or </em><code>valueLength</code><em> parameter
is negative an </em><code>ArrayIndexOutOfBoundsException</code>
         * <em> exception is thrown and no append is performed.</em>
         * <li><em>If </em><code>valueOffset+valueLength</code><em> is greater than
</em><code>value.length</code><em>, the length
         * of the </em><code>value</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
         * and no append is performed.</em>
         * </ul>
         *
         * @param tag the tag of the TLV to append, including the Comprehension Required flag
         * @param value the buffer containing the TLV value
         * @param valueOffset the offset of the TLV value in the buffer
         * @param valueLength the value length of the TLV to append
         *
         * @exception NullPointerException if <code>value</code> is <code>null</code>
         * @exception ArrayIndexOutOfBoundsException  ~~if <code>valueOffset</code> or
<code>valueLength</code>~~

     ~~    *        or both~~*       - if append would cause access of data outside array bounds~~, or if
<code>value2Length</code> is negative~~.
         * @exception ToolkitException with the following reason codes: <ul>
         *      <li><code>HANDLER_OVERFLOW</code> if the EditHandler buffer is to small to append the
requested data
         *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
         *      <li><code>BAD_INPUT_PARAMETER</code> if <code>valueLength</code> is greater than
255</ul>
         */
    public void appendTLV(byte tag,
                          byte[] value,
                          short valueOffset,
                          short valueLength) throws NullPointerException,
                                                    ArrayIndexOutOfBoundsException,
                                                    ToolkitException {
    }
```

```
     /**
     * Appends a TLV element to the current TLV list (1 byte and a byte array format).
     * A successful append does not modify the TLV selected.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>value2Offset</code><em> or </em><code>value2Length</code><em> parameter
is negative an </em><code>ArrayIndexOutOfBoundsException</code><em>
     * exception is thrown and no append is performed.</em>
     * <li><em>If </em><code>value2Offset+value2Length</code><em> is greater than
</em><code>value2.length</code><em>, the length
     * of the </em><code>value2</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no append is performed.</em>
     * </ul>
     *
     * @param tag the tag of the TLV to append, including the Comprehension Required flag
     * @param value1 the first byte in the value field
     * @param value2 the buffer containing the rest of the TLV field
     * @param value2Offset the offset of the rest of the TLV field in the buffer
     * @param value2Length the value length of the rest of the TLV field to append
     *
     * @exception NullPointerException if <code>value2</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException   ~~if <code>value2Offset</code> or
<code>value2Length</code>~~

     ~~*         or both~~*       - if append would cause access of data outside array bounds~~, or if
<code>value2Length</code> is negative~~.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_OVERFLOW</code> if the EditHandler buffer is to small to append the
requested data
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>BAD_INPUT_PARAMETER</code> if <code>value2Length</code> is greater than
254</ul>
     */
    public void appendTLV(byte tag,
                          byte value1,
                          byte[] value2,
                          short value2Offset,
                          short value2Length) throws NullPointerException,
                                                     ArrayIndexOutOfBoundsException,
                                                     ToolkitException {

    }
}
```

Class MEProfile

```
    /**
     * Checks a set of facilities in the handset profile.
     * The method checks all the facilities corresponding to bits set to 1 in
     * the mask buffer.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative
an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no check is performed.</em>
     * <li><em>If </em><code>offset+length</code><em> is greater than
</em><code>mask.length</code><em>, the length
     * of the </em><code>mask</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no check is performed.</em>
     * </ul>
     *
     * @param mask a byte array containing the mask to compare with the profile
     * @param offset the starting offset of the mask in the byte array
     * @param length the length of the mask (at least 1)
     *
     * @return true if the set of facilities is supported, false otherwise
     *
     * @exception NullPointerException if <code>mask</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>offset</code> or <code>length</code> or both~~* - if check would cause access ~~outside
array bounds~~of data outside mask array bounds.
```

```
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li>ME_PROFILE_NOT_AVAILABLE if Terminal Profile data are not available</ul>
     */
    public static boolean check(byte[] mask,
                                short offset,
                                short length) throws NullPointerException,
                                                     ArrayIndexOutOfBoundsException,
                                                     ToolkitException {
        return false;
    }
}
```

Class ProactiveHandler

```
    /**
     * Builds a Display Text Proactive command without sending the command. The Comprehension
     * Required flags are all set to 1.
     * After the method invocation no TLV is selected.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative
an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no proactive command is build.</em>
     * <li><em>If </em><code>offset+length</code><em> is greater than
</em><code>buffer.length</code><em>, the length
     * of the </em><code>buffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no proactive command is build.</em>
     * </ul>
         *
     * @param qualifier Display Text command qualifier
     * @param dcs data coding scheme
     * @param buffer reference to the text string source buffer
     * @param offset offset of the text string in the source buffer
     * @param length length of the text string in the source buffer
     *
     * @exception NullPointerException if <code>buffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException if <code>offset</code> or <code>length</code> or
both would cause access outside array bounds
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_OVERFLOW</code> if the ProactiveHandler buffer is to small to put the
requested data </ul>
     */
    public void initDisplayText(byte qualifier,
                                byte dcs,
                                byte[] buffer,
                                short offset,
                                short length) throws NullPointerException,
                                                     ArrayIndexOutOfBoundsException,
                                                     ToolkitException {
    }

    /**
     * Builds a Get Inkey Proactive command without sending the command. The Comprehension
     * Required flags are all set to 1.
     * After the method invocation no TLV is selected.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative
an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no proactive command is build.</em>
     * <li><em>If </em><code>offset+length</code><em> is greater than
</em><code>buffer.length</code><em>, the length
     * of the </em><code>buffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no proactive command is build.</em>
     * </ul>
         *
     * @param qualifier Get Inkey command qualifier
     * @param dcs data coding scheme
     * @param buffer reference to the displayed text string source buffer
     * @param offset offset of the displayed text string in the source buffer
     * @param length length of the displayed text string in the source buffer
     *
     * @exception NullPointerException if <code>buffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>offset</code> or <code>length</code> or both~~*           - if append would cause access
of data outside array bounds.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_OVERFLOW</code> if the ProactiveHandler buffer is to small to put the
requested data</ul>
     */
    public void initGetInkey(byte qualifier,
                             byte dcs,
                             byte[] buffer,
                             short offset,
                             short length) throws NullPointerException,
                                                  ArrayIndexOutOfBoundsException,
                                                  ToolkitException {
    }

    /**
     * Initialize the building of a Get Input Proactive command. The Comprehension
     * Required flags are all set to 1.
     * The following command parameters (i.e. TLVs) may be appended to the
     * command before sending it: Default Text.
     * After the method invocation no TLV is selected.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative
an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no proactive command is build.</em>
     * <li><em>If </em><code>offset+length</code><em>is greater than
</em><code>buffer.length</code><em>, the length
     * of the </em><code>buffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no proactive command is build.</em>
     * </ul>
         *
     * @param qualifier Get Input command qualifier
     * @param dcs data coding scheme
     * @param buffer reference to the displayed text string source buffer
     * @param offset offset of the displayed text string in the source buffer
     * @param length length of the displayed text string in the source buffer
     * @param minRespLength minimal length of the response text string
     * @param maxRespLength maximal length of the response text string
     *
     * @exception NullPointerException if <code>buffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>offset</code> or <code>length</code> or both~~*        - if append would cause access of
data outside array bounds.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_OVERFLOW</code> if the ProactiveHandler buffer is to small to put the
requested data</ul>
     */
    public void initGetInput(byte qualifier,
                             byte dcs,
                             byte[] buffer,
                             short offset,
                             short length,
                             short minRespLength,
                             short maxRespLength) throws NullPointerException,
                                                         ArrayIndexOutOfBoundsException,
                                                         ToolkitException {
    }
}
```

Class ProactiveResponseHandler

```
    /**
     * Copies a part of the additional information field from the first Result
     * TLV element of the current response data field.
     * If the element is available it becomes the TLV selected.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>dstOffset</code><em> or </em><code>dstLength</code><em> parameter is
negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no copy is performed.</em>
     * <li><em>If </em><code>dstOffset+dstLength</code><em> is greater than
</em><code>dstBuffer.length</code><em>, the length
     * of the </em><code>dstBuffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no copy is performed.</em>
     * </ul>
         *
     * @param dstBuffer a reference to the destination buffer
     * @param dstOffset the position in the destination buffer
     * @param dstLength the data length to be copied
     *
     * @return <code>dstOffset+dstLength</code>
     *
     * @exception NullPointerException if <code>dstBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException ~~if <code>dstOffset</code> or <code>dstLength</code>~~

     ~~*      or both~~*      - if append would cause access of data outside array bounds.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>UNAVAILABLE_ELEMENT</code> in case of unavailable Result TLV element
     ~~*      <li><code>OUT_OF_TLV_BOUNDARIES</code>~~*      <li><code>OUT_OF_TLV_BOUNDARIES</code>
if <code>dstLength</code> is greater than
         *      the value field of the available TLV</ul>
     */
    public short copyAdditionalInformation(byte[] dstBuffer,
                                        ~~short~~   short dstOffset,
                                        ~~short~~   short dstLength) throws NullPointerException,
                                                         ~~ArrayIndexOutOfBoundsException,~~
ArrayIndexOutOfBoundsException,
                                                         ~~ToolkitException~~
ToolkitException {
        return 0;
    }
```

Class ToolkitRegistry

```
    /**
     * Sets an event list in the Toolkit Registry entry of the applet.
     * In case of any exception the state of the registry is undefined. The toolkit applet has to
include this
         * call within a transaction if necessary.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative
an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no event list is set.</em>
     * <li><em>If </em><code>offset+length</code><em> is greater than
</em><code>eventList.length</code><em>, the length
     * of the </em><code>eventList</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no event list is set.</em>
     * </ul>
         *
     * @param eventList buffer containing the list of the new events to register
     * @param offset offset in the eventlist buffer for event registration
     * @param length length in the eventlist buffer for event registration
     *
     * @exception NullPointerException if <code>eventlist</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>offset</code> or <code>length</code> or both~~*   - if append would cause access of data
outside array bounds.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li>EVENT_NOT_SUPPORTED if one event is not supported
     *      <li>EVENT_ALREADY_REGISTERED if one event has already been registered
     *         (for limited event like Call Control)
     *    ~~<li>EVENT_NOT_ALLOWED~~     <li>EVENT_NOT_ALLOWED if eventList contains
EVENT_MENU_SELECTION,
     *      EVENT_MENU_SELECTION_HELP_REQUEST, EVENT_TIMER_EXPIRATION, EVENT_STATUS_COMMAND</ul>
     */
    public void setEventList(byte[] eventList,
                     ~~short~~     short offset,
                            ~~short~~     short length) throws NullPointerException,
                                                      ArrayIndexOutOfBoundsException,
                                                      ToolkitException {
    }


    /**
     * Initialises the next menu entry allocated at loading. The default state of the menu entry is
     * 'enabled'. The value of the <code>helpSupported</code> boolean parameter
     * defines the registration status of the applet to the event
     * EVENT_MENU_SELECTION_HELP_REQUEST. The applet is registered to
     * the EVENT_MENU_SELECTION. The icon identifier provided will be added to
     * the icon identifier list of the item icon identifier list Simple TLV if
     * all the applets registered to the EVENT_MENU_SELECTION provide it.
     * The Icon list qualifier transmitted to the ME will be 'icon is not self
     * explanatory' if one of the applet registered prefers this qualifier.
         * This method shall be called by the applet in the same order than the
         * order of the item parameters defined at the applet loading if the applet
         * has several menu entries. The applet shall initialise all its loaded
         * menu entries during its installation.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative
an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no menu entry is initialised.</em>
     * <li><em>If </em><code>offset+length</code><em> is greater than
</em><code>menuEntry.length</code><em>, the length
     * of the </em><code>menuEntry</code><em> array a
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no menu entry is initialised.</em>
     * </ul>
     *
     * @param menuEntry a reference on a byte array, containing the menu entry string
     * @param offset offset of the menu entry string in the buffer
     * @param length length of the menu entry string
     * @param nextAction a byte coding the next action indicator for the menu entry (or 0)
     * @param helpSupported equals true if help is available for the menu entry
     * @param iconQualifier the preferred value for the icon list qualifier
     * @param iconIdentifier the icon identifier for the menu entry  (0 means no icon)
     *
     * @return the identifier attached to the initialised menu entry
     *
     * @exception NullPointerException if <code>menuEntry</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>offset</code> or <code>length</code> or both~~*  - if append would cause access of data
outside array bounds. outside array bounds
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li>REGISTRY_ERROR if the menu entry cannot be initialised (eg no more item data in
applet loading parameter)
     ~~*      <li>ALLOWED_LENGTH_EXCEEDED~~*        <li>ALLOWED_LENGTH_EXCEEDED if the menu entry
string is bigger than the alloacted space</ul>
     */
    public byte initMenuEntry(byte[] menuEntry,
                              short offset,
                              short length,
                              byte nextAction,
                              boolean helpSupported,
                              byte iconQualifier,
                              short iconIdentifier) throws NullPointerException,
                                                           ArrayIndexOutOfBoundsException,
                                                           ToolkitException {
        return 0;
    }

    /**
     * Changes the value of a menu entry. The default state of the changed menu
     * entry is 'enabled'. The value of the <code>helpSupported</code> boolean
     * parameter defines the registration status of the EVENT_MENU_SELECTION_HELP_REQUEST
     * event. The icon identifier provided will be added to the icon identifier list
     * of the item icon identifier list Simple TLV if all the applets registered
     * to the EVENT_MENU_SELECTION provide it.
     * The Icon list qualifier transmitted to the ME will be 'icon is not self
     * explanatory' if one of the applet registered prefers this qualifier.
         * After the invocation of this method, during the current card session, the SIM Toolkit
Framework
         * shall dynamically update the menu stored in the ME.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative
an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no menu entry is changed.</em>
     * <li><em>If </em><code>offset+length</code><em> is greater than
</em><code>menuEntry.length</code><em>, the length
     * of the </em><code>menuEntry</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no menu entry is changed.</em>
     * </ul>
     *
     * @param id the menu entry identifier supplied by the <code>initMenuEntry()</code> method
     * @param menuEntry a reference on a byte array, containing the menu entry string
     * @param offset the position of the menu entry string in the buffer
     * @param length the length of the menu entry string
     * @param nextAction a byte coding the next action indicator for the menu entry (or 0)
     * @param helpSupported equals true if help is available for the menu entry
     * @param iconQualifier the preferred value for the icon list qualifier
     * @param iconIdentifier the icon identifier for the menu entry  (0 means no icon)
     *
     * @exception NullPointerException if <code>menuEntry</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>offset</code> or <code>length</code> or both~~*  - if append would cause access of data
outside array bounds.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li>MENU_ENTRY_NOT_FOUND if the menu entry does not exist for this applet
         *      <li>ALLOWED_LENGTH_EXCEEDED if the menu entry string is bigger than the alloacted
space</ul>
     */
    public void changeMenuEntry(byte id,
                                byte[] menuEntry,
                                short offset,
                                short length,
                                byte nextAction,
                                boolean helpSupported,
                                byte iconQualifier,
                                short iconIdentifier) throws NullPointerException,
                                                             ArrayIndexOutOfBoundsException,
                                                             ToolkitException {
    }
```

Class ViewHandler

```
    /**
     * Copies the simple TLV list contained in the handler to the destination byte array.
     *
     * @param dstBuffer a reference to the destination buffer
     * @param dstOffset the position in the destination buffer
     * @param dstLength the data length to be copied
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>dstOffset</code><em> or </em><code>dstLength</code><em> parameter is
negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no copy is performed.</em>
     * <li><em>If </em><code>dstOffset+dstLength</code><em> is greater than
</em><code>dstBuffer.length</code><em>, the length
     * of the </em><code>dstBuffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no copy is performed.</em>
     * </ul>
     *
     * @return <code>dstOffset+dstLength</code>
     *
     * @exception NullPointerException if <code>dstBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException      ~~if <code>dstOffset</code> or
<code>dstLength</code>~~

     ~~    *          or both~~*  - if append would cause access of data outside array bounds~~, or if
<code>dstLength</code> is negative~~.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>OUT_OF_TLV_BOUNDARIES</code> if <code>dstLength</code> is grater than the
length of the simple TLV List.</ul>
```

```
     */
    public short copy(  byte[] dstBuffer,
                        short dstOffset,
                        short dstLength) throws NullPointerException,
                                                ArrayIndexOutOfBoundsException,
                                                ToolkitException {
        return 0;
    }


    /**
     * Copies a part of the last TLV element which has been found, into a
     * destination buffer.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>dstOffset</code><em> or </em><code>dstLength</code><em> parameter is
negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no copy is performed.</em>
     * <li><em>If </em><code>dstOffset+dstLength</code><em> is greater than
</em><code>dstBuffer.length</code><em>, the length
     * of the </em><code>dstBuffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no copy is performed.</em>
     * </ul>
     *
     * @param valueOffset the offset of the first byte in the source TLV element
     * @param dstBuffer a reference to the destination buffer
     * @param dstOffset the position in the destination buffer
     * @param dstLength the data length to be copied
     *
     * @return <code>dstOffset+dstLength</code>
     *
     * @exception NullPointerException if <code>dstBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException      ~~if <code>dstOffset</code> or
<code>dstLength</code>~~

     ~~*       or both~~*   - if append would cause access of data outside array bounds~~, or if
<code>dstLength</code> is negative~~.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>UNAVAILABLE_ELEMENT</code> in case of unavailable TLV element
     *      <li><code>OUT_OF_TLV_BOUNDARIES</code> if <code>valueOffset</code>,
<code>dstLength</code> or both are out of the current TLV </ul>
     */
    public short copyValue( short valueOffset,
                            byte[] dstBuffer,
                            short dstOffset,
                            short dstLength) throws NullPointerException,
                                                    ArrayIndexOutOfBoundsException,
                                                    ToolkitException {
        return 0;
    }

    /**
     * Compares the last found TLV element with a buffer.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>compareOffset</code><em> or </em><code>compareLength</code><em>
parameter is negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no compare is performed.</em>
     * <li><em>If </em><code>compareOffset+compareLength</code><em>is greater than
</em><code>compareBuffer.length</code><em>, the length
     * of the </em><code>compareBuffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no compare is performed.</em>
     * </ul>
     *
     * @param valueOffset the offset of the first byte to compare in the TLV element
     * @param compareBuffer a reference to the comparison buffer
     * @param compareOffset the position in the comparison buffer
     * @param compareLength the length to be compared
     *
     * @return the result of the comparison as follows: <ul>
     *      <li><code>0</code> if identical
     *      <li><code>-1</code> if the first miscomparing byte in simple TLV List is less than that
in <code>compareBuffer</code>,
     *      <li><code>1</code> if the first miscomparing byte in simple TLV List is greater than
that in <code>compareBuffer</code>.</ul>
     *
     * @exception NullPointerException if <code>compareBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException      ~~if <code>compareOffset</code> or~~
~~<code>compareLength</code>~~

     ~~*        or both~~*   - if append would cause access of data outside array bounds~~, or if~~
~~<code>compareLength</code> is negative~~.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>UNAVAILABLE_ELEMENT</code> in case of unavailable TLV element
     *      <li><code>OUT_OF_TLV_BOUNDARIES</code> if <code>valueOffset</code>,
<code>compareLength</code> or both are out of the current TLV </ul>
     */
    public byte compareValue(short valueOffset,
                             byte[] compareBuffer,
                             short compareOffset,
                             short compareLength) throws    NullPointerException,
                                                            ArrayIndexOutOfBoundsException,
                                                            ToolkitException {
        return 0;
    }

    /**
     * Looks for the first occurence of a TLV element from the beginning of a TLV
     * list and copy its value into a destination buffer.
     * If no TLV element is found, the <code>UNAVAILABLE_ELEMENT</code> exception is thrown.
     * If the method is successful then the corresponding TLV becomes current,
     * else no TLV is selected.
     * This search method is Comprehension Required flag independent.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>dstOffset</code><em> parameter is negative or
</em><code>dstOffset</code>
     * <em> is greater than </em><code>dstBuffer.length</code><em>, the length of the
</em><code>dstBuffer</code>
     * <em> array an </em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown and no
find is performed.</em>
     * </ul>
     *
     * @param tag the tag of the TLV element to search
     * @param dstBuffer a reference to the destination buffer
     * @param dstOffset the position in the destination buffer
     *
     * @return <code>dstOffset</code> + length of the copied value
     *
     * @exception NullPointerException if <code>dstBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>dstOffset</code>~~*   - if append would cause access of data outside array bounds.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>UNAVAILABLE_ELEMENT</code> in case of unavailable TLV element</ul>
     */
    public short findAndCopyValue(byte tag,
                                  byte[] dstBuffer,
                                  short dstOffset) throws   NullPointerException,
                                                            ArrayIndexOutOfBoundsException,
                                                            ToolkitException {
        return 0;
    }

    /**
     * Looks for the indicated occurence of a TLV element from the beginning of a TLV
     * list and copy its value into a destination buffer.
     * If no TLV element is found, the <code>UNAVAILABLE_ELEMENT</code> exception is thrown.
     * If the method is successful then the corresponding TLV becomes current,
     * else no TLV is selected.
     * This search method is Comprehension Required flag independent.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>dstOffset</code><em> or </em><code>dstLength</code><em> parameter is
negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no copy is performed.</em>
     * <li><em>If </em><code>dstOffset+dstLength</code><em>is greater than
</em><code>dstBuffer.length</code><em>, the length
     * of the </em><code>dstBuffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no copy is performed.</em>
     * </ul>
     *
     * @param tag the tag of the TLV element to search
     * @param occurrence the occurrence number of the TLV element (1 for the first, 2 for the
second...)
     * @param valueOffset the offset of the first byte in the source TLV element
     * @param dstBuffer a reference to the destination buffer
     * @param dstOffset the position in the destination buffer
     * @param dstLength the data length to be copied
     *
     * @return <code>dstOffset + dstLength</code>
     *
     * @exception NullPointerException if <code>dstBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException      ~~if <code>dstOffset</code> or
<code>dstLength</code>~~

     ~~*           or both~~*  - if append would cause access of data outside array bounds~~, or if
<code>dstLength</code> is negative~~.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>UNAVAILABLE_ELEMENT</code> in case of unavailable TLV element
     *      <li><code>OUT_OF_TLV_BOUNDARIES</code> if <code>valueOffset</code>,
<code>dstLength</code> or both are out of the current TLV
     *      <li><code>BAD_INPUT_PARAMETER</code> if an input parameter is not valid (e.g. occurence
= 0)</ul>
     */
    public short findAndCopyValue(byte tag,
                                  byte occurence,
                                  short valueOffset,
                                  byte[] dstBuffer,
                                  short dstOffset,
                                  short dstLength) throws   NullPointerException,
                                                            ArrayIndexOutOfBoundsException,
                                                            ToolkitException {
        return 0;
    }

    /**
     * Looks for the first occurence of a TLV element from beginning of a TLV
     * list and compare its value with a buffer.
     * If no TLV element is found, the <code>UNAVAILABLE_ELEMENT</code> exception is thrown.
     * If the method is successful then the corresponding TLV becomes current,
     * else no TLV is selected.
     * This search method is Comprehension Required flag independent.
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>compareOffset</code><em> parameter is negative or
</em><code>compareOffset</code>
     * <em> is greater than </em><code>compareBuffer.length</code><em>, the length of the
</em><code>compareBuffer</code>
     * <em> array an </em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown and no
find is performed.</em>
     * </ul>
     *
     * @param tag the tag of the TLV element to search
     * @param compareBuffer a reference to the comparison buffer
     * @param compareOffset the position in the comparison buffer
     *
     * @return the result of the comparison as follows: <ul>
     *      <li><code>0</code> if identical
     *      <li><code>-1</code> if the first miscomparing byte in simple TLV is less than that in
<code>compareBuffer</code>,
     *      <li><code>1</code> if the first miscomparing byte in simple TLV is greater than that in
<code>compareBuffer</code>.</ul>
     *
     * @exception NullPointerException if <code>compareBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>compareOffset</code>~~*  - if append would cause access of data outside array bounds.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>UNAVAILABLE_ELEMENT</code> in case of unavailable TLV element</ul>
     */
    public byte findAndCompareValue(byte tag,
                                    byte[] compareBuffer,
                                    short compareOffset) throws NullPointerException,
                                                                ArrayIndexOutOfBoundsException,
                                                                ToolkitException {
        return 0;
    }

    /**
     * Looks for the indicated occurence of a TLV element from the beginning of a
     * TLV list and compare its value with a buffer.
     * If no TLV element is found, the <code>UNAVAILABLE_ELEMENT</code> exception is thrown.
     * If the method is successful then the corresponding TLV becomes current,
     * else no TLV is selected.
     * This search method is Comprehension Required flag independent.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>compareOffset</code><em> or </em><code>compareLength</code><em>
parameter is negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no find and compare is performed.</em>
     * <li><em>If </em><code>compareOffset+compareLength</code><em> is greater than
</em><code>compareBuffer.length</code><em>, the length
     * of the </em><code>compareBuffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no find and compare is performed.</em>
     * </ul>
     *
     * @param tag the tag of the TLV element to search
     * @param occurrence the occurrence number of the TLV element (1 for the first, 2 for the
second...)
     * @param valueOffset the offset of the first byte in the source TLV element
     * @param compareBuffer a reference to the comparison buffer
     * @param compareOffset the position in the comparison buffer
     * @param compareLength the length to be compared
     *
     * @return the result of the comparison as follows: <ul>
     *      <li><code>0</code> if identical
     *      <li><code>-1</code> if the first miscomparing byte in simple TLV is less than that in
<code>compareBuffer</code>,
     *      <li><code>1</code> if the first miscomparing byte in simple TLV is greater than that in
<code>compareBuffer</code>.</ul>
     *
     * @exception NullPointerException if <code>compareBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException      ~~if <code>compareOffset</code> or
<code>compareLength</code>~~

     ~~*       or both~~*   - if append would cause access of data outside array bounds~~, or if
<code>compareLength</code> is negative~~.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>UNAVAILABLE_ELEMENT</code> in case of unavailable TLV element
     *      <li><code>OUT_OF_TLV_BOUNDARIES</code> if <code>valueOffset</code>,
<code>compareLength</code> or both are out of the current TLV
     *      <li><code>BAD_INPUT_PARAMETER</code> if an input parameter is not valid (e.g. occurence
= 0)</ul>
     */
    public byte findAndCompareValue(byte tag,
                                    byte occurence,
                                    short valueOffset,
                                    byte[] compareBuffer,
                                    short compareOffset,
                                    short compareLength) throws NullPointerException,
                                                                ArrayIndexOutOfBoundsException,
                                                                ToolkitException {
        return 0;
    }
}
```

3GPP TSG-T3 Meeting #18 ***Tdoc T3-010252***
Sofia Antipolis, France, 1st - 2nd of March 2001

CR-Form-v3

CHANGE REQUEST

⌘	**03.19**	**CR**	**011**	⌘	rev	**-**	⌘	Current version:	8.0.0	⌘

For ***HELP*** *on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM [✓ X] ME/UE ☐ Radio Access Network ☐ Core Network ☐

Title:	⌘	Clarification about ArrayIndexOutOfBoundsException		
Source:	⌘	T3		
Work item code:	⌘		***Date:*** ⌘	02/03/2001
Category:	⌘	**A**	***Release:*** ⌘	R99

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	Clarify the reason for an ArrayIndexOutOfBoundException
Summary of change:	⌘	For all methods throwing an ArrayIndexOutOfBoundException introduction of a Note that explains in which case this exception is thrown
Consequences if not approved:	⌘	Can cause interoperability problems with different implementations

Clauses affected:	⌘	Annex_A_java	
Other specs affected:	⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘
Other comments:	⌘		

How to create CRs using this form:

Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://www.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2000-09 contains the specifications resulting from the September 2000 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

Package sim.access

Class SIMView

```
    /**
     * SELECT command as defined in GSM 11.11 standard.<br>
     * By default, the MF is selected at the beginning of each applet activation
     * (current file = MF). This method selects a file of the GSM file system.
     * The file search starts at the current DF of the applet according to
     * the file search method described in GSM 11.11 specification. The current
     * DF or current EF and the current record pointer of the calling applet will
     * be changed after successful execution.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>fciOffset</code><em> or </em><code>fciLength</code><em> parameter is
negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no read is performed.</em>
     * <li><em>If </em><code>fciOffset+fciLength</code><em> is greater than
</em><code>fci.length</code><em>, the length
     * of the </em><code>fci</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no status request is performed.</em>
     * </ul>
     *
     * @param fid is the File Identifier of the file to be selected.
     * @param fci is the reference to the target byte array for FCI (File Control
     *        Information) of current file, coding is according to GSM 11.11.
     *        If <code>fci</code> is <code>null</code> the <code>NullPointerException</code> is
thrown.
     * @param fciOffset is the offset in the <code>fci</code> buffer for the response data.
     * @param fciLength is the length of the required data in the <code>fci</code> byte array.
     *        If the <code>fciLength</code> is greater than the length of the response, the whole
     *        response is copied into the <code>fci</code> buffer and the length of the response
     *            is returned by the method. If the <code>fcilength</code> is smaller than the
length
     *        of the response, the first part of the response is copied into the <code>fci</code>
     *            buffer and the <code>fciLength</code> is returned by the method.
     *
     * @return length of the data which have been written in the <code>fci</code> buffer
     *         (cannot be greater than <code>fciLength</code> parameter)
     *
     * @exception NullPointerException if <code>fci</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException if operation would cause access of data outside
     *         array bounds.
     * @exception SIMViewException in case of error<ul>
     *          <li><code>FILE_NOT_FOUND</code>
     *          <li><code>MEMORY_PROBLEM</code>
     *          <li><code>INTERNAL_ERROR</code></ul>
     */
    public short select(short fid,
                        byte  fci[],
                        short fciOffset,
                        short fciLength) throws NullPointerException,
                                                ArrayIndexOutOfBoundsException,
                                                SIMViewException;


    /**
     * STATUS command as defined in GSM 11.11 standard.<br>
     * This method returns the FCI (File Control Information) of the current DF
     * (or MF) of the calling applet.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>fciOffset</code><em> or </em><code>fciLength</code><em> parameter is
negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no read is performed.</em>
     * <li><em>If </em><code>fciOffset+fciLength</code><em> is greater than
</em><code>fci.length</code><em>, the length
     * of the </em><code>fci</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no status request is performed.</em>
     * </ul>
     *
     * @param fci is the reference to the target byte array for FCI (File Control
     *        Information) of current DF (or MF), coding is according to GSM 11.11.
     *        If <code>fci</code> is <code>null</code> the <code>NullPointerException</code> is
thrown.
     * @param fciOffset is the offset in the <code>fci</code> buffer for the response data.
     * @param fciLength is the length of the required data in the <code>fci</code> byte array.
         *        If the <code>fciLength</code> is greater than the length of the response, the
whole
         *        response is copied into the <code>fci</code> buffer and the length of the response
         *            is returned by the method. If the <code>fcilength</code> is smaller than the
length
         *        of the response, the first part of the response is copied into the
<code>fci</code>
         *            buffer and the <code>fciLength</code> is returned by the method.
     *
     * @return length of the data which have been written in the <code>fci</code> buffer
     *         (cannot be greater than <code>fciLength</code> parameter)
     *
     * @exception NullPointerException if <code>fci</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException if operation would cause access of data outside
         *         array bounds.
     * @exception SIMViewException in case of error <ul>
     *          <li><code>MEMORY_PROBLEM</code>
     *          <li><code>INTERNAL_ERROR</code></ul>
     */
    public short status(byte  fci[],
                        short fciOffset,
                        short fciLength) throws NullPointerException,
                                                ArrayIndexOutOfBoundsException,
                                                SIMViewException;

    /**
     * READ BINARY command as defined in GSM 11.11 standard.<br>
     * This method reads the data bytes of the current transparent EF of the
     * calling applet.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>respOffset</code><em> or </em><code>respLength</code><em> parameter is
negative a </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no read is performed.</em>
     * <li><em>If </em><code>respOffset+respLength</code><em>is greater than
</em><code>resp.length</code><em>, the length
     * of the </em><code>resp</code><em> array a
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no select is performed.</em>
     * </ul>
     *
     * @param fileOffset is the offset in the source transparent file.
     * @param resp is the reference to the response byte array for read data.
~~     *        If <code>resp</code> is <code>null</code> the <code>NullPointerException</code> is~~
~~thrown.~~
~~     *        If <code>respOffset</code> or <code>respLength</code> or~~
~~<code>respOffset+respLength</code> is in contradiction~~
~~     *        with the <code>resp</code> byte array the <code>ArrayIndexOutOfBoundsException</code>~~
~~is thrown.~~
     * @param respOffset is the offset in the response byte array.
     * @param respLength is the number of bytes to read.
     *
     * @return <code>respOffset+respLength</code>
     *
     * @exception NullPointerException if <code>resp</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     *      if reading would cause access of data outside array bounds
     * @exception SIMViewException in case of error <ul>
     *          <li><code>NO_EF_SELECTED</code>
     *          <li><code>FILE_INCONSISTENT</code>
     *          <li><code>AC_NOT_FULFILLED</code>
     *          <li><code>INVALIDATION_STATUS_CONTRADICTION</code>
     *          <li><code>OUT_OF_FILE_BOUNDARIES</code>
     *          <li><code>MEMORY_PROBLEM</code>
     *          <li><code>INTERNAL_ERROR</code></ul>
     */
    public short readBinary(short fileOffset,
                            byte  resp[],
                            short respOffset,
                            short respLength) throws NullPointerException,
                                                     ArrayIndexOutOfBoundsException,
                                                     SIMViewException;
    /**
     * UPDATE BINARY command as defined in GSM 11.11 standard.<br>
     * This method updates the data bytes of the current transparent EF of
     * the calling applet.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>dataOffset</code><em> or </em><code>dataLength</code><em> parameter is
negative a </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no update is performed.</em>
     * <li><em>If </em><code>dataOffset+dataLength</code><em> is greater than
</em><code>data.length</code><em>, the length
     * of the </em><code>data</code><em> array a
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no update is performed.</em>
     * </ul>
     *
     * @param fileOffset is the offset in the destination transparent file.
     * @param data is the reference to the source byte array for data to update.
~~     *         If <code>data</code> is <code>null</code> the <code>NullPointerException</code> is
thrown.~~
~~     *         If <code>dataOffset</code> or <code>dataLength</code> or
<code>dataOffset+dataLength</code> is in contradiction~~
~~     *         with the <code>data</code> byte array the <code>ArrayIndexOutOfBoundsException</code>
is thrown.~~
     * @param dataOffset is the offset in the source byte array.
     * @param dataLength is the number of bytes to update.
     *
     * @exception NullPointerException if <code>data</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     *      if updating would cause access of data outside array bounds
     * @exception SIMViewException in case of error<ul>
     *         <li><code>NO_EF_SELECTED</code>
     *         <li><code>FILE_INCONSISTENT</code>
     *         <li><code>AC_NOT_FULFILLED</code>
     *         <li><code>INVALIDATION_STATUS_CONTRADICTION</code>
     *         <li><code>OUT_OF_FILE_BOUNDARIES</code>
     *         <li><code>MEMORY_PROBLEM</code>
     *         <li><code>INTERNAL_ERROR</code></ul>
     */
    public void updateBinary(short fileOffset,
                             byte  data[],
                             short dataOffset,
                             short dataLength) throws   NullPointerException,
                                                        ArrayIndexOutOfBoundsException,
                                                        SIMViewException;

    /**
     * READ RECORD command as defined in GSM 11.11 standard.<br>
     * This method reads the data bytes of the current linear fixed/cyclic EF
     * of the calling applet. The current record pointer can be changed due to
     * the choosen mode.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>respOffset</code><em> or </em><code>respLength</code><em> parameter is
negative a </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no read is performed.</em>
     * <li><em>If </em><code>respOffset+respLength</code><em> is greater than
</em><code>resp.length</code><em>, the length
     * of the </em><code>resp</code><em> array a
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no select is performed.</em>
     * </ul>
     *
     * @param recNumber is the record number.
     * @param mode is the mode for reading record, according to GSM 11.11.
     *         If <code>mode</code> is <code>REC_ACC_MODE_NEXT</code> and the record pointer is at
the last record
         *         the <code>RECORD_NUMBER_NOT_AVAILABLE SIMViewException</code> shall be thrown.
         *         If <code>mode</code> is <code>REC_ACC_MODE_PREVIOUS</code> and the record pointer
is at the first record,
         *         the <code>RECORD_NUMBER_NOT_AVAILABLE SIMViewException</code> shall be thrown.
     * @param recOffset is the offset in the record for the data to read.
     * @param resp is the reference to the response byte array for read data.
~~     *         If <code>resp</code> is <code>null</code> the <code>NullPointerException</code> is
thrown.~~
```

```
~~     *        If <code>respOffset</code> or <code>respLength</code> or~~
~~<code>respOffset+respLength</code> is in contradiction~~
~~     *        with the <code>resp</code> byte array the <code>ArrayIndexOutOfBoundsException</code>~~
~~is thrown.~~
     * @param respOffset is the offset in the response byte array.
     * @param respLength is the number of bytes to read.
     *
     * @return <code>respOffset+respLength</code>
     *
     * @exception NullPointerException if <code>resp</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     *      if reading would cause access of data outside array bounds
     * @exception SIMViewException in case of error <ul>
     *         <li><code>NO_EF_SELECTED</code>
     *         <li><code>FILE_INCONSISTENT</code>
     *         <li><code>AC_NOT_FULFILLED</code>
     *         <li><code>INVALIDATION_STATUS_CONTRADICTION</code>
     *         <li><code>OUT_OF_RECORD_BOUNDARIES</code>
     *         <li><code>RECORD_NUMBER_NOT_AVAILABLE</code>
     *         <li><code>INVALID_MODE</code>
     *         <li><code>MEMORY_PROBLEM</code>
     *         <li><code>INTERNAL_ERROR</code></ul>
     */
    public short readRecord(short recNumber,
                            byte  mode,
                            short recOffset,
                            byte  resp[],
                            short respOffset,
                            short respLength) throws     NullPointerException,
                                                         ArrayIndexOutOfBoundsException,
                                                         SIMViewException;

    /**
     * UPDATE RECORD command as defined in GSM 11.11 standard.<br>
     * This method updates the data bytes of the current linear fixed/cyclic EF
     * of the calling applet. The current record pointer can be changed due to
     * the choosen mode.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>dataOffset</code><em> or </em><code>dataLength</code><em> parameter is
negative a </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no select is performed.</em>
     * <li><em>If </em><code>dataOffset+dataLength</code><em> is greater than
</em><code>data.length</code><em>, the length
     * of the </em><code>data</code><em> array a
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no update is performed.</em>
     * </ul>
     *
     * @param recNumber is the record number.
     * @param mode is the mode for updating record, according to GSM 11.11.
     *        If <code>mode</code> is <code>REC_ACC_MODE_NEXT</code> and the record pointer is at
the last record
         *        the <code>RECORD_NUMBER_NOT_AVAILABLE SIMViewException</code> shall be thrown.
         *        If <code>mode</code> is <code>REC_ACC_MODE_PREVIOUS</code> and the record pointer
is at the first record,
         *        the <code>RECORD_NUMBER_NOT_AVAILABLE SIMViewException</code> shall be thrown.
     * @param recOffset is the offset in the record for the data to update.
     * @param data is the reference to the source byte array for data to update.
~~     *        If <code>data</code> is <code>null</code> the <code>NullPointerException</code> is~~
~~thrown.~~
~~     *        If <code>dataOffset</code> or <code>dataLength</code> or~~
~~<code>dataOffset+dataLength</code> is in contradiction~~
~~     *        with the <code>data</code> byte array the <code>ArrayIndexOutOfBoundsException</code>~~
~~is thrown.~~
     * @param dataOffset is the offset in the source byte array.
     * @param dataLength is the number of bytes to update.
     *
     * @exception NullPointerException if <code>data</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     *       if updating would cause access of data outside array bounds
     * @exception SIMViewException in case of error<ul>
     *         <li><code>NO_EF_SELECTED</code>
     *         <li><code>FILE_INCONSISTENT</code>
     *         <li><code>AC_NOT_FULFILLED</code>
     *         <li><code>INVALIDATION_STATUS_CONTRADICTION</code>
     *         <li><code>OUT_OF_RECORD_BOUNDARIES</code>
     *          <li><code>RECORD_NUMBER_NOT_AVAILABLE</code>
     *          <li><code>INVALID_MODE</code>
     *          <li><code>MEMORY_PROBLEM</code>
     *          <li><code>INTERNAL_ERROR</code></ul>
     */
    public void updateRecord(short recNumber,
                             byte  mode,
                             short recOffset,
                             byte  data[],
                             short dataOffset,
                             short dataLength) throws   NullPointerException,
                                                        ArrayIndexOutOfBoundsException,
                                                        SIMViewException;

    /**
     * SEEK command as defined in GSM 11.11 standard.<br>
     * This method seeks a pattern in the current linear fixed EF of the calling
     * applet.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>pattOffset</code><em> or </em><code>pattLength</code><em> parameter is
negative a </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no seek is performed.</em>
     * <li><em>If </em><code>pattOffset+pattLength</code><em> is greater than
</em><code>patt.length</code><em>, the length
     * of the </em><code>patt</code><em> array a
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no seek is performed.</em>
     * </ul>
     *
     * @param mode is the seek mode, according to GSM 11.11 (no type information).
     *          If <code>mode</code> is <code>SEEK_FROM_NEXT_FORWARD</code> and the record pointer is
at the last record
     *          the <code>PATTERN_NOT_FOUND SIMViewException</code> shall be thrown.
     *          If <code>mode</code> is <code>SEEK_FROM_PREVIOUS_BACKWARD</code> and the record
pointer is at the first record,
     *          the <code>PATTERN_NOT_FOUND SIMViewException</code> shall be thrown.
     * @param patt is the reference to the byte array containing the seek pattern.
~~     *          If <code>patt</code> is <code>null</code> the <code>NullPointerException</code> is
thrown.~~
~~     *          If <code>pattOffset</code> or <code>pattLength</code> or
<code>pattOffset+pattLength</code> is in contradiction~~
~~     *          with the <code>patt</code> byte array the <code>ArrayIndexOutOfBoundsException</code>
is thrown.~~
     * @param pattOffset is the offset of the seek pattern in the byte array.
     * @param pattLength is the length of the seek pattern.
     *          If <code>pattLength</code> is greater than the current record size than
     *          the <code>OUT_OF_RECORD_BOUNDARIES SIMViewException</code> shall be thrown.
     *          If <code>pattLength</code> is <code>zero</code> than <code>PATTERN_NOT_FOUND
SIMViewException</code> shall be thrown.
     *
     * @return record number if pattern found
     *
     * @exception NullPointerException if <code>patt</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     *      if seeking would cause access of data outside array bounds
     * @exception SIMViewException in case of error<ul>
     *          <li><code>NO_EF_SELECTED</code>
     *          <li><code>PATTERN_NOT_FOUND</code>
     *          <li><code>FILE_INCONSISTENT</code>
     *          <li><code>AC_NOT_FULFILLED</code>
     *          <li><code>INVALIDATION_STATUS_CONTRADICTION</code>
     *          <li><code>INVALID_MODE</code>
     *          <li><code>OUT_OF_RECORD_BOUNDARIES</code>
     *          <li><code>MEMORY_PROBLEM</code>
     *          <li><code>INTERNAL_ERROR</code></ul>
     */
    public short seek(byte   mode,
                      byte[] patt,
                      short  pattOffset,
                      short  pattLength) throws  NullPointerException,
                                                ArrayIndexOutOfBoundsException,
                                                SIMViewException;

    /**
     * INCREASE command as defined in GSM 11.11 standard.<br>
     * This method increases the current cyclic EF record of the calling applet.
     * The response buffer will only contain the value of the increased record.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>incrOffset</code><em> or </em><code>respOffset</code><em> parameter is
negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no increase is performed.</em>
     * <li><em>If </em><code>incrOffset</code><em> is greater than
</em><code>incr.length</code><em>, the length
     * of the </em><code>incr</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no increase is performed.</em>
     * <li><em>If </em><code>respOffset</code><em> is greater than
</em><code>resp.length</code><em>, the length
     * of the </em><code>resp</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no increase is performed.</em>
     * </ul>
     *
     * @param incr is the reference to the source byte array, containing the
     *        value to add, on 3 bytes.
     * @param incrOffset is the offset in the source byte array.
     * @param resp is the reference to the response byte array for new record value.
~~*        If <code>incr</code> or <code>resp</code> is <code>null</code> the
<code>NullPointerException</code> is thrown.~~
     *        If <code>resp</code> buffer is smaller than the record size, the
<code>ArrayIndexOutOfBoundsException</code> is thrown.
     *        If <code>resp</code> buffer is bigger than the record size, the <code>resp</code>
buffer is filled
     *        with the record value and left justified
     * @param respOffset is the offset in the response byte array.
     *
     * @return length of the valid data in the <code>resp</code> buffer
     *      (cannot be greater than the record size)
     *
     * @exception NullPointerException if <code>incr</code> or <code>resp</code> is
<code>null</code>
     * @exception ArrayIndexOutOfBoundsException if increasing would cause access of data outside
array bounds
     * @exception SIMViewException in case of error<ul>
     *         <li><code>NO_EF_SELECTED</code>
     *         <li><code>FILE_INCONSISTENT</code>
     *         <li><code>AC_NOT_FULFILLED</code>
     *         <li><code>INVALIDATION_STATUS_CONTRADICTION</code>
     *         <li><code>MAX_VALUE_REACHED</code>
     *         <li><code>MEMORY_PROBLEM</code>
     *         <li><code>INTERNAL_ERROR</code></ul>
     */
    public short increase(byte[] incr,
                          short  incrOffset,
                          byte[] resp,
                          short  respOffset) throws  NullPointerException,
                                                     ArrayIndexOutOfBoundsException,
                                                     SIMViewException;
```

Package sim.toolkit

Class EditHandler

```
    /**
     * Clears the TLV list of an EditHandler and resets the current TLV selected.
     *
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy</ul>
     */
    public void clear() throws ToolkitException {
    }

    /**
     * Appends a buffer into the EditHandler buffer.
         * A successful append does not modify the TLV selected.
         * The TLV list structure of the handler should be maintained by the applet in the
         * appended array (e.g. the length of the TLV element should be coded according to ISO 7816-
6),
         * if the TLV manipulation methods are to be used afterwards with the handler.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative
an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no append is performed.</em>
     * <li><em>If </em><code>offset+length</code><em> is greater than
</em><code>buffer.length</code><em>, the length
     * of the </em><code>buffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no append is performed.</em>
     * </ul>
     *
     * @param buffer the buffer containing data for copy
     * @param offset the offset in the buffer
     * @param length the value length of the buffer
     *
     * @exception NullPointerException if <code>buffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException ~~if <code>offset</code> or <code>length</code>~~

     ~~*          or both~~*  - if append would cause access of data outside array bounds~~, or if~~
~~<code>length</code> is negative~~.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_OVERFLOW</code> if the EditHandler buffer is to small to append the
requested data
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy</ul>
     */
    public void appendArray(byte[] buffer,
                            short offset,
                            short length) throws NullPointerException,
                                                 ArrayIndexOutOfBoundsException,
                                                 ToolkitException {
    }

    /**
     * Appends a TLV element to the current TLV list (byte array format).
     * A successful append does not modify the TLV selected.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>valueOffset</code><em> or </em><code>valueLength</code><em> parameter
is negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no append is performed.</em>
     * <li><em>If </em><code>valueOffset+valueLength</code><em> is greater than
</em><code>value.length</code><em>, the length
     * of the </em><code>value</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no append is performed.</em>
     * </ul>
     *
     * @param tag the tag of the TLV to append, including the Comprehension Required flag
     * @param value the buffer containing the TLV value
     * @param valueOffset the offset of the TLV value in the buffer
     * @param valueLength the value length of the TLV to append
     *
     * @exception NullPointerException if <code>value</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException   ~~if <code>valueOffset</code> or~~
~~<code>valueLength</code>~~

     ~~*          or both~~*      - if append would cause access of data outside array bounds~~, or if~~
~~<code>value2Length</code> is negative~~.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_OVERFLOW</code> if the EditHandler buffer is to small to append the
requested data
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>BAD_INPUT_PARAMETER</code> if <code>valueLength</code> is greater than
255</ul>
     */
    public void appendTLV(byte tag,
                          byte[] value,
                          short valueOffset,
                          short valueLength) throws NullPointerException,
                                                    ArrayIndexOutOfBoundsException,
                                                    ToolkitException {
    }
```

```
     /**
     * Appends a TLV element to the current TLV list (1 byte and a byte array format).
     * A successful append does not modify the TLV selected.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>value2Offset</code><em> or </em><code>value2Length</code><em> parameter is negative an </em><code>ArrayIndexOutOfBoundsException</code><em>
     * exception is thrown and no append is performed.</em>
     * <li><em>If </em><code>value2Offset+value2Length</code><em> is greater than </em><code>value2.length</code><em>, the length
     * of the </em><code>value2</code><em> array an </em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no append is performed.</em>
     * </ul>
     *
     * @param tag the tag of the TLV to append, including the Comprehension Required flag
     * @param value1 the first byte in the value field
     * @param value2 the buffer containing the rest of the TLV field
     * @param value2Offset the offset of the rest of the TLV field in the buffer
     * @param value2Length the value length of the rest of the TLV field to append
     *
     * @exception NullPointerException if <code>value2</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException   ~~if <code>value2Offset</code> or <code>value2Length</code>~~

     ~~*       or both~~*       - if append would cause access of data outside array bounds~~, or if <code>value2Length</code> is negative~~.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_OVERFLOW</code> if the EditHandler buffer is to small to append the requested data
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>BAD_INPUT_PARAMETER</code> if <code>value2Length</code> is greater than 254</ul>
     */
    public void appendTLV(byte tag,
                          byte value1,
                          byte[] value2,
                          short value2Offset,
                          short value2Length) throws NullPointerException,
                                                     ArrayIndexOutOfBoundsException,
                                                     ToolkitException {
    }
}
```

Class MEProfile

```
    /**
     * Checks a set of facilities in the handset profile.
     * The method checks all the facilities corresponding to bits set to 1 in
     * the mask buffer.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no check is performed.</em>
     * <li><em>If </em><code>offset+length</code><em> is greater than </em><code>mask.length</code><em>, the length
     * of the </em><code>mask</code><em> array an </em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no check is performed.</em>
     * </ul>
     *
     * @param mask a byte array containing the mask to compare with the profile
     * @param offset the starting offset of the mask in the byte array
     * @param length the length of the mask (at least 1)
     *
     * @return true if the set of facilities is supported, false otherwise
     *
     * @exception NullPointerException if <code>mask</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>offset</code> or <code>length</code> or both~~* - if check would cause access ~~outside array bounds~~of data outside mask array bounds.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li>ME_PROFILE_NOT_AVAILABLE if Terminal Profile data are not available</ul>
     */
    public static boolean check(byte[] mask,
                                short offset,
                                short length) throws NullPointerException,
                                                     ArrayIndexOutOfBoundsException,
                                                     ToolkitException {
        return false;
    }
}
```

Class ProactiveHandler

```
    /**
     * Builds a Display Text Proactive command without sending the command. The Comprehension
     * Required flags are all set to 1.
     * After the method invocation no TLV is selected.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative
an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no proactive command is build.</em>
     * <li><em>If </em><code>offset+length</code><em> is greater than
</em><code>buffer.length</code><em>, the length
     * of the </em><code>buffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no proactive command is build.</em>
     * </ul>
         *
     * @param qualifier Display Text command qualifier
     * @param dcs data coding scheme
     * @param buffer reference to the text string source buffer
     * @param offset offset of the text string in the source buffer
     * @param length length of the text string in the source buffer
     *
     * @exception NullPointerException if <code>buffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException if <code>offset</code> or <code>length</code> or
both would cause access outside array bounds
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_OVERFLOW</code> if the ProactiveHandler buffer is to small to put the
requested data </ul>
     */
    public void initDisplayText(byte qualifier,
                                byte dcs,
                                byte[] buffer,
                                short offset,
                                short length) throws NullPointerException,
                                                     ArrayIndexOutOfBoundsException,
                                                     ToolkitException {
    }

    /**
     * Builds a Get Inkey Proactive command without sending the command. The Comprehension
     * Required flags are all set to 1.
     * After the method invocation no TLV is selected.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative
an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no proactive command is build.</em>
     * <li><em>If </em><code>offset+length</code><em> is greater than
</em><code>buffer.length</code><em>, the length
     * of the </em><code>buffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no proactive command is build.</em>
     * </ul>
         *
     * @param qualifier Get Inkey command qualifier
     * @param dcs data coding scheme
     * @param buffer reference to the displayed text string source buffer
     * @param offset offset of the displayed text string in the source buffer
     * @param length length of the displayed text string in the source buffer
     *
     * @exception NullPointerException if <code>buffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>offset</code> or <code>length</code> or both~~*           - if append would cause access
of data outside array bounds.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_OVERFLOW</code> if the ProactiveHandler buffer is to small to put the
requested data</ul>
     */
    public void initGetInkey(byte qualifier,
                             byte dcs,
                             byte[] buffer,
                             short offset,
                             short length) throws NullPointerException,
                                                  ArrayIndexOutOfBoundsException,
                                                  ToolkitException {
    }

    /**
     * Initialize the building of a Get Input Proactive command. The Comprehension
     * Required flags are all set to 1.
     * The following command parameters (i.e. TLVs) may be appended to the
     * command before sending it: Default Text.
     * After the method invocation no TLV is selected.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative
an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no proactive command is build.</em>
     * <li><em>If </em><code>offset+length</code><em>is greater than
</em><code>buffer.length</code><em>, the length
     * of the </em><code>buffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no proactive command is build.</em>
     * </ul>
         *
     * @param qualifier Get Input command qualifier
     * @param dcs data coding scheme
     * @param buffer reference to the displayed text string source buffer
     * @param offset offset of the displayed text string in the source buffer
     * @param length length of the displayed text string in the source buffer
     * @param minRespLength minimal length of the response text string
     * @param maxRespLength maximal length of the response text string
     *
     * @exception NullPointerException if <code>buffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>offset</code> or <code>length</code> or both~~*        - if append would cause access of
data outside array bounds.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_OVERFLOW</code> if the ProactiveHandler buffer is to small to put the
requested data</ul>
     */
    public void initGetInput(byte qualifier,
                             byte dcs,
                             byte[] buffer,
                             short offset,
                             short length,
                             short minRespLength,
                             short maxRespLength) throws NullPointerException,
                                                         ArrayIndexOutOfBoundsException,
                                                         ToolkitException {
    }
}
```

Class ProactiveResponseHandler

```
    /**
     * Copies a part of the additional information field from the first Result
     * TLV element of the current response data field.
     * If the element is available it becomes the TLV selected.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>dstOffset</code><em> or </em><code>dstLength</code><em> parameter is
negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no copy is performed.</em>
     * <li><em>If </em><code>dstOffset+dstLength</code><em> is greater than
</em><code>dstBuffer.length</code><em>, the length
     * of the </em><code>dstBuffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no copy is performed.</em>
     * </ul>
         *
     * @param dstBuffer a reference to the destination buffer
     * @param dstOffset the position in the destination buffer
     * @param dstLength the data length to be copied
     *
     * @return <code>dstOffset+dstLength</code>
     *
     * @exception NullPointerException if <code>dstBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException ~~if <code>dstOffset</code> or <code>dstLength</code>~~

     ~~*       or both~~*      - if append would cause access of data outside array bounds.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>UNAVAILABLE_ELEMENT</code> in case of unavailable Result TLV element
     ~~*       <li><code>OUT_OF_TLV_BOUNDARIES</code>~~*      <li><code>OUT_OF_TLV_BOUNDARIES</code>
if <code>dstLength</code> is greater than
         *       the value field of the available TLV</ul>
     */
    public short copyAdditionalInformation(byte[] dstBuffer,
                                          ~~short~~   short dstOffset,
                                          ~~short~~   short dstLength) throws NullPointerException,
                                                                        ~~ArrayIndexOutOfBoundsException,~~
ArrayIndexOutOfBoundsException,
                                                                        ~~ToolkitException~~
ToolkitException {
        return 0;
    }
```

Class ToolkitRegistry

```
    /**
     * Sets an event list in the Toolkit Registry entry of the applet.
     * In case of any exception the state of the registry is undefined. The toolkit applet has to
include this
         * call within a transaction if necessary.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative
an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no event list is set.</em>
     * <li><em>If </em><code>offset+length</code><em> is greater than
</em><code>eventList.length</code><em>, the length
     * of the </em><code>eventList</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no event list is set.</em>
     * </ul>
         *
     * @param eventList buffer containing the list of the new events to register
     * @param offset offset in the eventlist buffer for event registration
     * @param length length in the eventlist buffer for event registration
     *
     * @exception NullPointerException if <code>eventlist</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>offset</code> or <code>length</code> or both~~*  - if append would cause access of data
outside array bounds.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li>EVENT_NOT_SUPPORTED if one event is not supported
     *      <li>EVENT_ALREADY_REGISTERED if one event has already been registered
     *         (for limited event like Call Control)
     *    ~~<li>EVENT_NOT_ALLOWED~~    <li>EVENT_NOT_ALLOWED if eventList contains
EVENT_MENU_SELECTION,
     *      EVENT_MENU_SELECTION_HELP_REQUEST, EVENT_TIMER_EXPIRATION, EVENT_STATUS_COMMAND</ul>
     */
    public void setEventList(byte[] eventList,
                            ~~short~~     short offset,
                              ~~short~~      short length) throws NullPointerException,
                                                    ArrayIndexOutOfBoundsException,
                                                    ToolkitException {
    }


    /**
     * Initialises the next menu entry allocated at loading. The default state of the menu entry is
     * 'enabled'. The value of the <code>helpSupported</code> boolean parameter
     * defines the registration status of the applet to the event
     * EVENT_MENU_SELECTION_HELP_REQUEST. The applet is registered to
     * the EVENT_MENU_SELECTION. The icon identifier provided will be added to
     * the icon identifier list of the item icon identifier list Simple TLV if
     * all the applets registered to the EVENT_MENU_SELECTION provide it.
     * The Icon list qualifier transmitted to the ME will be 'icon is not self
     * explanatory' if one of the applet registered prefers this qualifier.
         * This method shall be called by the applet in the same order than the
         * order of the item parameters defined at the applet loading if the applet
         * has several menu entries. The applet shall initialise all its loaded
         * menu entries during its installation.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative
an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no menu entry is initialised.</em>
     * <li><em>If </em><code>offset+length</code><em> is greater than
</em><code>menuEntry.length</code><em>, the length
     * of the </em><code>menuEntry</code><em> array a
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no menu entry is initialised.</em>
     * </ul>
     *
     * @param menuEntry a reference on a byte array, containing the menu entry string
     * @param offset offset of the menu entry string in the buffer
     * @param length length of the menu entry string
     * @param nextAction a byte coding the next action indicator for the menu entry (or 0)
     * @param helpSupported equals true if help is available for the menu entry
     * @param iconQualifier the preferred value for the icon list qualifier
     * @param iconIdentifier the icon identifier for the menu entry  (0 means no icon)
     *
     * @return the identifier attached to the initialised menu entry
     *
     * @exception NullPointerException if <code>menuEntry</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>offset</code> or <code>length</code> or both~~*   - if append would cause access of data
outside array bounds. outside array bounds
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li>REGISTRY_ERROR if the menu entry cannot be initialised (eg no more item data in
applet loading parameter)
     ~~*      <li>ALLOWED_LENGTH_EXCEEDED~~*         <li>ALLOWED_LENGTH_EXCEEDED if the menu entry
string is bigger than the alloacted space</ul>
     */
    public byte initMenuEntry(byte[] menuEntry,
                              short offset,
                              short length,
                              byte nextAction,
                              boolean helpSupported,
                              byte iconQualifier,
                              short iconIdentifier) throws NullPointerException,
                                                           ArrayIndexOutOfBoundsException,
                                                           ToolkitException {
        return 0;
    }

    /**
     * Changes the value of a menu entry. The default state of the changed menu
     * entry is 'enabled'. The value of the <code>helpSupported</code> boolean
     * parameter defines the registration status of the EVENT_MENU_SELECTION_HELP_REQUEST
     * event. The icon identifier provided will be added to the icon identifier list
     * of the item icon identifier list Simple TLV if all the applets registered
     * to the EVENT_MENU_SELECTION provide it.
     * The Icon list qualifier transmitted to the ME will be 'icon is not self
     * explanatory' if one of the applet registered prefers this qualifier.
         * After the invocation of this method, during the current card session, the SIM Toolkit
Framework
         * shall dynamically update the menu stored in the ME.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>offset</code><em> or </em><code>length</code><em> parameter is negative
an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no menu entry is changed.</em>
     * <li><em>If </em><code>offset+length</code><em> is greater than
</em><code>menuEntry.length</code><em>, the length
     * of the </em><code>menuEntry</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no menu entry is changed.</em>
     * </ul>
     *

     * @param id the menu entry identifier supplied by the <code>initMenuEntry()</code> method
     * @param menuEntry a reference on a byte array, containing the menu entry string
     * @param offset the position of the menu entry string in the buffer
     * @param length the length of the menu entry string
     * @param nextAction a byte coding the next action indicator for the menu entry (or 0)
     * @param helpSupported equals true if help is available for the menu entry
     * @param iconQualifier the preferred value for the icon list qualifier
     * @param iconIdentifier the icon identifier for the menu entry  (0 means no icon)
     *
     * @exception NullPointerException if <code>menuEntry</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>offset</code> or <code>length</code> or both~~*   - if append would cause access of data
outside array bounds.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li>MENU_ENTRY_NOT_FOUND if the menu entry does not exist for this applet
         *      <li>ALLOWED_LENGTH_EXCEEDED if the menu entry string is bigger than the alloacted
space</ul>
     */
    public void changeMenuEntry(byte id,
                                byte[] menuEntry,
                                short offset,
                                short length,
                                byte nextAction,
                                boolean helpSupported,
                                byte iconQualifier,
                                short iconIdentifier) throws NullPointerException,
                                                             ArrayIndexOutOfBoundsException,
                                                             ToolkitException {
    }
```

Class ViewHandler

```
    /**
     * Copies the simple TLV list contained in the handler to the destination byte array.
     *
     * @param dstBuffer a reference to the destination buffer
     * @param dstOffset the position in the destination buffer
     * @param dstLength the data length to be copied
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>dstOffset</code><em> or </em><code>dstLength</code><em> parameter is
negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no copy is performed.</em>
     * <li><em>If </em><code>dstOffset+dstLength</code><em> is greater than
</em><code>dstBuffer.length</code><em>, the length
     * of the </em><code>dstBuffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no copy is performed.</em>
     * </ul>
     *
     * @return <code>dstOffset+dstLength</code>
     *
     * @exception NullPointerException if <code>dstBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException      ~~if <code>dstOffset</code> or~~
~~<code>dstLength</code>~~

     ~~    *        or both~~*   - if append would cause access of data outside array bounds~~, or if~~
~~<code>dstLength</code> is negative~~.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>OUT_OF_TLV_BOUNDARIES</code> if <code>dstLength</code> is grater than the
length of the simple TLV List.</ul>
```

```
     */
    public short copy(  byte[] dstBuffer,
                        short dstOffset,
                        short dstLength) throws NullPointerException,
                                                ArrayIndexOutOfBoundsException,
                                                ToolkitException {
        return 0;
    }


    /**
     * Copies a part of the last TLV element which has been found, into a
     * destination buffer.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>dstOffset</code><em> or </em><code>dstLength</code><em> parameter is
negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no copy is performed.</em>
     * <li><em>If </em><code>dstOffset+dstLength</code><em> is greater than
</em><code>dstBuffer.length</code><em>, the length
     * of the </em><code>dstBuffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no copy is performed.</em>
     * </ul>
     *
     * @param valueOffset the offset of the first byte in the source TLV element
     * @param dstBuffer a reference to the destination buffer
     * @param dstOffset the position in the destination buffer
     * @param dstLength the data length to be copied
     *
     * @return <code>dstOffset+dstLength</code>
     *
     * @exception NullPointerException if <code>dstBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException      ~~if <code>dstOffset</code> or~~
~~<code>dstLength</code>~~

     ~~*      or both~~*   - if append would cause access of data outside array bounds~~, or if~~
~~<code>dstLength</code> is negative~~.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>UNAVAILABLE_ELEMENT</code> in case of unavailable TLV element
     *      <li><code>OUT_OF_TLV_BOUNDARIES</code> if <code>valueOffset</code>,
<code>dstLength</code> or both are out of the current TLV </ul>
     */
    public short copyValue( short valueOffset,
                            byte[] dstBuffer,
                            short dstOffset,
                            short dstLength) throws NullPointerException,
                                                    ArrayIndexOutOfBoundsException,
                                                    ToolkitException {
        return 0;
    }

    /**
     * Compares the last found TLV element with a buffer.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>compareOffset</code><em> or </em><code>compareLength</code><em>
parameter is negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no compare is performed.</em>
     * <li><em>If </em><code>compareOffset+compareLength</code><em>is greater than
</em><code>compareBuffer.length</code><em>, the length
     * of the </em><code>compareBuffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no compare is performed.</em>
     * </ul>
     *
     * @param valueOffset the offset of the first byte to compare in the TLV element
     * @param compareBuffer a reference to the comparison buffer
     * @param compareOffset the position in the comparison buffer
     * @param compareLength the length to be compared
     *
     * @return the result of the comparison as follows: <ul>
     *      <li><code>0</code> if identical
     *      <li><code>-1</code> if the first miscomparing byte in simple TLV List is less than that
in <code>compareBuffer</code>,
     *      <li><code>1</code> if the first miscomparing byte in simple TLV List is greater than
that in <code>compareBuffer</code>.</ul>
     *
     * @exception NullPointerException if <code>compareBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException      ~~if <code>compareOffset</code> or
<code>compareLength</code>~~

     ~~    *       or both~~*   - if append would cause access of data outside array bounds~~, or if
<code>compareLength</code> is negative~~.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>UNAVAILABLE_ELEMENT</code> in case of unavailable TLV element
     *      <li><code>OUT_OF_TLV_BOUNDARIES</code> if <code>valueOffset</code>,
<code>compareLength</code> or both are out of the current TLV </ul>
     */
    public byte compareValue(short valueOffset,
                             byte[] compareBuffer,
                             short compareOffset,
                             short compareLength) throws    NullPointerException,
                                                            ArrayIndexOutOfBoundsException,
                                                            ToolkitException {
        return 0;
    }

    /**
     * Looks for the first occurence of a TLV element from the beginning of a TLV
     * list and copy its value into a destination buffer.
     * If no TLV element is found, the <code>UNAVAILABLE_ELEMENT</code> exception is thrown.
     * If the method is successful then the corresponding TLV becomes current,
     * else no TLV is selected.
     * This search method is Comprehension Required flag independent.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>dstOffset</code><em> parameter is negative or
</em><code>dstOffset</code>
     * <em> is greater than </em><code>dstBuffer.length</code><em>, the length of the
</em><code>dstBuffer</code>
     * <em> array an </em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown and no
find is performed.</em>
     * </ul>
     *
     * @param tag the tag of the TLV element to search
     * @param dstBuffer a reference to the destination buffer
     * @param dstOffset the position in the destination buffer
     *
     * @return <code>dstOffset</code> + length of the copied value
     *
     * @exception NullPointerException if <code>dstBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>dstOffset</code>~~*   - if append would cause access of data outside array bounds.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>UNAVAILABLE_ELEMENT</code> in case of unavailable TLV element</ul>
     */
    public short findAndCopyValue(byte tag,
                                  byte[] dstBuffer,
                                  short dstOffset) throws   NullPointerException,
                                                            ArrayIndexOutOfBoundsException,
                                                            ToolkitException {
        return 0;
    }

    /**
     * Looks for the indicated occurence of a TLV element from the beginning of a TLV
     * list and copy its value into a destination buffer.
     * If no TLV element is found, the <code>UNAVAILABLE_ELEMENT</code> exception is thrown.
     * If the method is successful then the corresponding TLV becomes current,
     * else no TLV is selected.
     * This search method is Comprehension Required flag independent.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>dstOffset</code><em> or </em><code>dstLength</code><em> parameter is
negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no copy is performed.</em>
     * <li><em>If </em><code>dstOffset+dstLength</code><em>is greater than
</em><code>dstBuffer.length</code><em>, the length
     * of the </em><code>dstBuffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no copy is performed.</em>
     * </ul>
     *
     * @param tag the tag of the TLV element to search
     * @param occurrence the occurrence number of the TLV element (1 for the first, 2 for the
second...)
     * @param valueOffset the offset of the first byte in the source TLV element
     * @param dstBuffer a reference to the destination buffer
     * @param dstOffset the position in the destination buffer
     * @param dstLength the data length to be copied
     *
     * @return <code>dstOffset + dstLength</code>
     *
     * @exception NullPointerException if <code>dstBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException      ~~if <code>dstOffset</code> or
<code>dstLength</code>~~

     ~~      *        or both~~* - if append would cause access of data outside array bounds~~, or if
<code>dstLength</code> is negative~~.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>UNAVAILABLE_ELEMENT</code> in case of unavailable TLV element
     *      <li><code>OUT_OF_TLV_BOUNDARIES</code> if <code>valueOffset</code>,
<code>dstLength</code> or both are out of the current TLV
     *      <li><code>BAD_INPUT_PARAMETER</code> if an input parameter is not valid (e.g. occurence
= 0)</ul>
     */
    public short findAndCopyValue(byte tag,
                                  byte occurence,
                                  short valueOffset,
                                  byte[] dstBuffer,
                                  short dstOffset,
                                  short dstLength) throws   NullPointerException,
                                                            ArrayIndexOutOfBoundsException,
                                                            ToolkitException {
        return 0;
    }

    /**
     * Looks for the first occurence of a TLV element from beginning of a TLV
     * list and compare its value with a buffer.
     * If no TLV element is found, the <code>UNAVAILABLE_ELEMENT</code> exception is thrown.
     * If the method is successful then the corresponding TLV becomes current,
     * else no TLV is selected.
     * This search method is Comprehension Required flag independent.
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>compareOffset</code><em> parameter is negative or
</em><code>compareOffset</code>
     * <em> is greater than </em><code>compareBuffer.length</code><em>, the length of the
</em><code>compareBuffer</code>
     * <em> array an </em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown and no
find is performed.</em>
     * </ul>
     *
     * @param tag the tag of the TLV element to search
     * @param compareBuffer a reference to the comparison buffer
     * @param compareOffset the position in the comparison buffer
     *
     * @return the result of the comparison as follows: <ul>
     *      <li><code>0</code> if identical
     *      <li><code>-1</code> if the first miscomparing byte in simple TLV is less than that in
<code>compareBuffer</code>,
     *      <li><code>1</code> if the first miscomparing byte in simple TLV is greater than that in
<code>compareBuffer</code>.</ul>
     *
     * @exception NullPointerException if <code>compareBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException
     ~~if <code>compareOffset</code>~~* - if append would cause access of data outside array bounds.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>UNAVAILABLE_ELEMENT</code> in case of unavailable TLV element</ul>
     */
    public byte findAndCompareValue(byte tag,
                                    byte[] compareBuffer,
                                    short compareOffset) throws NullPointerException,
                                                                ArrayIndexOutOfBoundsException,
                                                                ToolkitException {
        return 0;
    }

    /**
     * Looks for the indicated occurence of a TLV element from the beginning of a
     * TLV list and compare its value with a buffer.
     * If no TLV element is found, the <code>UNAVAILABLE_ELEMENT</code> exception is thrown.
     * If the method is successful then the corresponding TLV becomes current,
     * else no TLV is selected.
     * This search method is Comprehension Required flag independent.
     *
     * <p>
     * Notes:<ul>
     * <li><em>If </em><code>compareOffset</code><em> or </em><code>compareLength</code><em>
parameter is negative an </em><code>ArrayIndexOutOfBoundsException</code>
     * <em> exception is thrown and no find and compare is performed.</em>
     * <li><em>If </em><code>compareOffset+compareLength</code><em> is greater than
</em><code>compareBuffer.length</code><em>, the length
     * of the </em><code>compareBuffer</code><em> array an
</em><code>ArrayIndexOutOfBoundsException</code><em> exception is thrown
     * and no find and compare is performed.</em>
     * </ul>
     *
     * @param tag the tag of the TLV element to search
     * @param occurrence the occurrence number of the TLV element (1 for the first, 2 for the
second...)
     * @param valueOffset the offset of the first byte in the source TLV element
     * @param compareBuffer a reference to the comparison buffer
     * @param compareOffset the position in the comparison buffer
     * @param compareLength the length to be compared
     *
     * @return the result of the comparison as follows: <ul>
     *      <li><code>0</code> if identical
     *      <li><code>-1</code> if the first miscomparing byte in simple TLV is less than that in
<code>compareBuffer</code>,
     *      <li><code>1</code> if the first miscomparing byte in simple TLV is greater than that in
<code>compareBuffer</code>.</ul>
     *
     * @exception NullPointerException if <code>compareBuffer</code> is <code>null</code>
     * @exception ArrayIndexOutOfBoundsException      ~~if <code>compareOffset</code> or
<code>compareLength</code>~~

     ~~*       or both~~*   - if append would cause access of data outside array bounds~~, or if
<code>compareLength</code> is negative~~.
     * @exception ToolkitException with the following reason codes: <ul>
     *      <li><code>HANDLER_NOT_AVAILABLE</code> if the handler is busy
     *      <li><code>UNAVAILABLE_ELEMENT</code> in case of unavailable TLV element
     *      <li><code>OUT_OF_TLV_BOUNDARIES</code> if <code>valueOffset</code>,
<code>compareLength</code> or both are out of the current TLV
     *      <li><code>BAD_INPUT_PARAMETER</code> if an input parameter is not valid (e.g. occurence
= 0)</ul>
     */
    public byte findAndCompareValue(byte tag,
                                    byte occurence,
                                    short valueOffset,
                                    byte[] compareBuffer,
                                    short compareOffset,
                                    short compareLength) throws NullPointerException,
                                                                ArrayIndexOutOfBoundsException,
                                                                ToolkitException {
        return 0;
    }
}
```