Function	Description
`<text>.GET_SIGNED8(<n>`	Treats the string of bytes represented by text as a sequence of 8-bit signed integers and returns the integer at byte offset n. If the offset makes all or part of the value outside of the current text, an UNDEFcondition is raised.
`<text>.GET_UNSIGNED8(<n>)`	Treats the string of bytes represented by text as a sequence of 8-bit unsigned integers and returns the integer at byte offset n. If the offset makes all or part of the value outside of the current text, an UNDEFcondition is raised.
`<text>.GET_SIGNED16(<n>, <endianness>)`	Treats the text string returned by the prefix as a string of bytes, extracts 16 bits starting at byte offset n, and converts the extracted bit sequence to a 16-bit signed integer. If the offset makes all or part of the value outside of the current text, an UNDEF condition is raised. The first parameter n is the byte offset from the current position in the text string. Providing a byte offset enables the function to handle items that are not aligned on the boundaries that are required by indexes. The second parameter, endianness, takes a mnemonic value of LITTLE_ENDIAN or BIG_ENDIAN. Note: In NetScaler 9.2, the parameter n was an index into an array of 16-bit items. In NetScaler 9.3, the parameter is a byte offset. Therefore, if you used this function in NetScaler 9.2, after you upgrade to NetScaler 9.3, you must change n to 2*n to obtain the same results as you did earlier. For example, if the value of n before the upgrade was 4, you must change the value of n to 8. The parameter endianness also no longer takes the values that it did in NetScaler 9.2, which were 0 and 1. Instead, endianness accepts the mnemonic values mentioned earlier. Example: HTTP.REQ.BODY(100).GET_SIGNED16(8, BIG_ENDIAN).
`<text>.GET_UNSIGNED16(<n>, <endianness>)`	Treats the text string returned by the prefix as a string of bytes, extracts 16 bits starting at byte offset n, and converts the extracted bit sequence to a 16-bit unsigned integer. If the offset makes all or part of the value outside of the current text, an UNDEF condition is raised. The first parameter n is the byte offset from the current position in the text string. Providing a byte offset enables the function to handle items that are not aligned on the boundaries that are required by indexes. The second

	parameter, endianness, takes a mnemonic value of LITTLE_ENDIAN or BIG_ENDIAN. Note: In NetScaler 9.2, the parameter n was an index into an array of 16-bit items. In NetScaler 9.3, the parameter is a byte offset. Therefore, if you used this function in NetScaler 9.2, after you upgrade to NetScaler 9.3, you must change n to 2*n to obtain the same results as you did earlier. For example, if the value of n before the upgrade was 4, you must change the value of n to 8. The parameter endianness also no longer takes the values that it did in NetScaler 9.2, which were 0 and 1. Instead, endianness accepts the mnemonic values mentioned earlier. Example: HTTP.REQ.BODY(100).GET_UNSIGNED16(8, LITTLE_ENDIAN)
<text>.GET_SIGNED32(<n>, <endianness>)	Treats the text string returned by the prefix as a string of bytes, extracts 32 bits starting at byte offset n, and converts the extracted bit sequence to a 32-bit signed integer. If the offset makes all or part of the value outside of the current text, an UNDEF condition is raised. The first parameter n is the byte offset from the current position in the text string. Providing a byte offset enables the function to handle items that are not aligned on the boundaries that are required by indexes. The second parameter, endianness, takes a mnemonic value of LITTLE_ENDIAN or BIG_ENDIAN. Note: In NetScaler 9.2, the parameter n was an index into an array of 32-bit items. In NetScaler 9.3, the parameter is a byte offset. Therefore, if you used this function in NetScaler 9.2, after you upgrade to NetScaler 9.3, you must change n to 4*n to obtain the same results as you did earlier. For example, if the value of n before the upgrade was 4, you must change the value of n to 16. The parameter endianness also no longer takes the values that it did in NetScaler 9.2, which were 0 and 1. Instead, endianness accepts the mnemonic values mentioned earlier. Example: HTTP.REQ.BODY(1000).GET_SIGNED32(12, BIG_ENDIAN)
<text>.GET_UNSIGNED32(<n>, <endianness>)	Treats the text string returned by the prefix as a string of bytes, extracts 32 bits starting at byte offset n, and returns the extracted bit sequence as part of a 64-bit unsigned long integer. If the offset makes all or part of the value outside of the current text, an UNDEFcondition is raised. The first parameter n is the byte offset from the current position in the text string. Providing a byte offset enables the function to handle items that are not

	aligned on the boundaries that are required by indexes. The second parameter, endianness, takes a mnemonic value of LITTLE_ENDIAN or BIG_ENDIAN. Example: HTTP.REQ.BODY(1000).GET_UNSIGNED32(30, LITTLE_ENDIAN