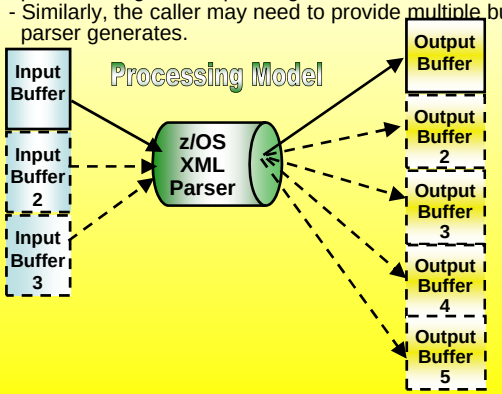
CheatSheet #35 zTidBits z10 zOS' XML Services

- z/OS XML System Services (z/OS XML) parser is a system level XML parser that is *integrated* with the base z/OS operating system and eligible for parsing operations to run on a **zAAP** engine .
  - It is intended for use by system components, middleware, and applications that need a simple, efficient, non-validating XML parsing solution.
    NOTE: A non-validating parser checks if a document is well-formed, but doe snot check a document against any DTDs or XML schemas. A validating parser not only checks if a document is well-formed, but also verifies that it conforms to a specific Document Type Definition (DTD) or XML Schema.
- The z/OS XML parser is invoked as a callable service and can be used as such.
  - The callable service stubs are shipped in zOS' SYS1.CSSLIB.
  - By default, the z/OS 'linklist' begins with SYS1.LINKLIB, SYS1.MIGLIB, and SYS1.CSSLIB.
- The following are some **distinct characteristics** of the z/OS XML parser:
  - The z/OS XML parser is an integrated parser for z/OS. There is no need to download or install any additional packages to use the z/OS XML parser.
  - The z/OS XML parser provides C/C++ and assembler interfaces for callers to use.
  - A caller can access these services through the z/OS XML System Services APIs.
  - The z/OS XML parser provides a buffer-in, buffer-out processing model instead of the event driven model common to SAX parsers.
    NOTE: The document to parse is provided by the caller in one buffer, and the z/OS XML parser creates a parsed record stream in another buffer which is also provided by the caller. The source document and the parsed data stream may actually each span one or more buffers, allowing the caller to feed the document to the z/OS XML parser in multiple pieces as well as receive output from the parser in the same fashion.
  - The z/OS XML parser uses **buffer spanning** to handle documents of unbounded length.
    > **Buffer spanning** enables the z/OS XML parser to use multiple buffers to contain the document being parsed, along with the parsed data stream generated from it.
  - The z/OS XML parser has minimal linkage overhead.
  - The z/OS XML parser provides assistive aids to the user in debugging not well formed documents.
  - The z/OS XML parser natively handles (no need to transcode into or from another encoding) a number of character encodings, among them UTF-8, UTF-16 (big endian), IBM-1047, and IBM-037.
- The z/OS XML System Services contains **two primary functions**: one for *querying* XML documents and another for *parsing* XML documents. These functions are provided in the form of the call services.
  - Querying XML documents XML documents contain characteristics that can affect their ability to be parsed. One such characteristic is the encoding scheme of the document, which the z/OS XML parser must know before parsing. Using the query service will allow the caller to acquire this information, after which it can then pass it to the z/OS XML parser.
    The z/OS XML parser will then be able to use the correct encoding scheme to parse the document.
  - Parsing XML documents The parsing process consists of three fundamental steps:
    a. initiate the parse process, b. parse the document, c. terminate the parse process.
    NOTE: The parse process can be repeated for multiple documents.
- There are **three main components** required for parsing XML documents:
  1) input buffer, 2) z/OS XML parser, 3) output buffer.
  - These three components and their interrelationships make up the processing model.
  - There may be more than one input and output buffer depending on the document being parsed.
  - If the document is sufficiently large, the caller may find it necessary to provide it to the parser in several pieces, using buffer spanning to maintain the document structure as it is being parsed.
  - Similarly, the caller may need to provide multiple buffers to contain the data stream that the z/OS XML parser generates.



Document processing is the creation of the output buffers from the parsed input data. As the z/OS XML parser traverses through the input buffer, the output buffer is built.

The left diagram displays the processing model using buffer spanning. It shows both the input and output buffers, where buffers 2-5 represent the additional buffers created to support a large document.

- The following is a list of features provided by z/OS XML System Services.
  - An external C and C++ API,
  - An external assembler API
  - Supports AMODE 31 and 64-bit callers with data above or below the bar
  - Support for UTF-8, UTF-16 (big endian only), IBM-1047, IBM-037, as well as several other encodings.
  - XML processing features
    > Support for XML 1.0 and XML 1.1
    > Newline normalization
    > Attribute value normalization
    > Returns comments
    > Returns white space
    > Namespace support
    > Entity resolution
    > Partial DTD processing
    > User exits for system services
    > Query service for determining docum
    > Diagnostic support
    > Slip trap support
    > ARR recovery routine
    > Segmented input & output (the entire document does not have to reside in a single buffer
    > Mapping macro interfaces for parsed data
    > zAAP enablement

z/OS XML provides an ARR recovery routine to assist with problem determination and diagnostics. This recovery routine can be turned o through an initialization option when invoked through the assembler API. For callers of the z/OS XL C/C++ parser API (gxlpParse), when running in Language Environment, the ARR recover routine is provided by default in most cases. For C or C++ callers who are running in either SRB mode or under an existing FRR routine, the z/OS XML ARR will not be provided, as it would not work properly in those environments.

NOTE: z/OS' XML System Services provides the ability for parsing operations to be run on a zAAP processor. The z/OS XML parser, when executing in TCB mode, is eligible to run on a zAAP, in environments in which one or more zAAPs are configured. Ancillary z/OS XML System Services, such as the query service and the control service, and memory management exits, are not eligible to run on a zAAP. Execution of z/OS XML System Services parsing operations on a zAAP occurs transparently to the calling application.

- Before the z/OS XML parser can parse an XML document, it must first establish a context in which it can operate.
  - This is accomplished when the caller invokes the initialization routine and passes in a piec of memory where the z/OS XML parser sets up a **Parse Instance Memory Area (PIMA)**.
  - This is the area where the z/OS XML parser creates a base for the internal data structures it uses to complete the parse process.
    Rule1: A particular PIMA must only be used during the parse of a single XML document at a time. Only after the parse is complete and the parse instance is reset can a PIMA be reused for the parse of another document. In addition to control informatio the PIMA is used as a memory area storing temporary data during the parse. When the z/OS XML parser needs more storage than was provided in the PIMA, additional storage is allocated. Because allocating additional storage is an expensiv operation, the PIMA should be initially allocated with sufficient storage to handle the expected document size, in order to optimize memory allocation requests.
    Rule2: The minimum size for the PIMA is 128k. Everything that the z/OS XML parser needs to complete the parse of a document is kept in the PIMA, along with any associated memory extensions that the parser may allocate during the process.
- The following steps **summarize parsing** XML documents using the z/OS XML parser:
  1. Call the initialization service. This establishes the PIMA, which is then used to create and store the initial data structures required to begin the parse process.
  2. Call the parse service to parse the document.
     Note: During the parse process and before the end of the document is reached, if the input buffer is empty or the output buffer is full, a warning is issued and the parse service is stopped. Otherwise, the parse service will continue until the document is fully processed
  3. The application processes the output buffer.
  4. Determine if there are additional documents to be processed. If so, call the termination service to terminate the existing parse process, and repeat Steps 1-3.
     Tip: For increased performance, the caller can use the control service in place of the termination and initialization services. The control service enables the PIMA to be reused, avoiding the need to free resources and re-initiate a new PIMA. However, the PIMA can only be reused in this way when the XML documents are in the same encoding.

Usage Considerations:

- z/OS XML System Services supports several code pages.
  -The caller must supply the CCSID of the encoding for the document at the time the z/OS XML parser is initialized.
- For callers that do not provide a memory allocation exit, the z/OS XML parser provides default routines to allocate and free memory. The z/OS XML parser also provides an option at initialization time allowing the caller to specify how the z/OS XML parser's default routine allocates memory.