# Speedbase Memory Database Manager (SMDM)

## 1. Introduction

This note describes the Speedbase Memory Database Manager (SMDM). SMDM permits database I/O operations to take place without a physical database. All I/O operations, which are always local to the current process, are directed to memory instead. Other documentation may refer to SMDM as the "Speedbase Access Method", SMAM or SMEM. The term "Speedbase Memory Database Manager" is preferred because the interface described here has no connection with a traditional Access Method.

Two versions of SMDM have been released. Restricted SMDM (R-SMDM) was released with GSM V8.1l. Extended SMDM (E-SMDM) was released with GSM SP-15.

## 2. Restricted SMDM (R-SMDM)

This section describes the initial release of the SMDM interface.

### 2.1 Database Definition

The restricted version of SMDM (R-SMDM) uses dictionaries as generated by the existing (and unmodified) $SDM Dictionary Maintenance utility, which allows up to 36 different record layouts to be defined. Each SMDM database is identified by a database ID beginning with "$$" followed by up to 3 further characters. Many different SMDM databases can therefore be specified.

The database specification process is unchanged. However, it is a restriction of R-SMDM that no Indexes or Master/Servant relationships can be defined for any record type. Note that $SDM has **not** been modified to validate this restriction, but a compilation error will result if an invalid record declaration is referenced in a program.

### 2.2 I/O Operations

All declarations and I/O statements used are standard Speedbase. An ACCESS statement is used to create an I/O channel to the required records, and the normal Speedbase I/O verbs are then used to perform I/O. Note that because index definitions are not permitted, you cannot make use of the FETCH and READ verbs (which always perform I/O via an index). Thus all I/O must be performed using the WRITE, REWRITE, DELETE and GET verbs.

### 2.3 Memory Management

All memory management is automatic. A 32Kb page of memory is automatically allocated to each R-SMDM I/O channel when the first I/O operation takes place, and further pages are subsequently allocated as required. Memory used by the I/O channel is automatically released when the program containing the I/O channel terminates. Since there is no physical database, there isn't anything to open. Hence there is no need to call B$OPN prior to performing I/O operations.

If necessary, the pages associated with an I/O channel can be explicitly cleared using the following call:

```
CALL B$SMDE USING $rt
```

where $*rt* is the name of the R-SMDM I/O channel. This call is functionally similar to deleting all records held within the I/O channel, but is much more memory efficient (as the actual memory pages are de-allocated). When loading a new record set into an I/O channel, any previous records

should therefore be cleared using this call.

## 2.4 Window Manager

Speedbase Window manager has been enhanced to work with R-SMDM databases, and all normal operations are supported.

## 2.5 Locking

In order to maintain consistency, the standard Speedbase locking facilities have been implemented in R-SMDM even though the process is inherently single user. A record must be locked before it is rewritten, however, a lock request can obviously never fail.

## 2.6 Special considerations

Each R-SMDM I/O channel operates in its own right. This means that if a duplicate I/O channel is ACCESS'ed, then each channel will operate independently (i.e. each channel will have it's own memory pages associated with it, in which a separate record set is stored).

## 2.7 Lookup capability

R-SMDM allows Enquiry Lookups for use by the Speedbase Window Manager only. This facility is intended to allow Read-only windows to perform look-ups on pre-sorted lists.

In order to make use of this facility, the R-SMDM record is defined with a Primary Index Definition only. **No secondary indexes may be specified, nor may there be any Master/ Servant declarations**. The application program must **NOT** attempt to perform I/O operations using the index, as this will lead to unpredictable errors.

When coding the Window definition, it is important to ensure that the window operates in ENQ and SEL modes only, by disabling both ADD and INS modes. MNT mode is permissible, provided that the primary index segments are coded as display only, so that they cannot be modified.

When writing records to the I/O channel (using the usual WRITE statement), you must write the records *in primary index order.* No actual index handling is performed by R-SMDM, and so it is totally up to you to ensure that the key order is properly maintained within the R-SMDM file. Note also that R-SMEM will **not** detect a Duplicate Primary Key situation using the Write operation.

In summary, this facility is intended purely as a look-up facility. You write the records in the correct key order, and invoke a Window. The normal window facilities can then be used to perform enquiries via the primary key.

## 2.8 Programming Notes

Restricted SMDM (R-SMDM) requires less memory and is slightly faster than Extended SMDM (E-SMDM). Therefore, R-SMDM is most suitable for processing simple lists, which are already in sorted order, and would therefore require no indexes. R-SMDM should be used in preference to E-SMDM in such cases. However R-SMDM is **not** suitable for more complex structures when indexes and master/servant relationships are required.

# 3. Extended SMDM (E-SMDM)

This section describes the Extended SMDM interface.

## 3.1 Database Definition

The extended version of SMDM (E-SMDM) uses dictionaries as generated by the existing (and unmodified) $DXM Dictionary Maintenance utility, which allows up to 512 different record layouts to be defined. Each E-SMDM database is identified by a database ID beginning with "$$" followed by up to 3 further characters. Many different SMDM databases can therefore be specified.

The database specification process is unchanged. You may define indexes and master/servant relationships as normal. However, it is a restriction of this version of E-SMDM that no cascading GVF's may be defined. Note that $DXM has **not** been modified to validate this restriction.

## 3.2 I/O Operations

All declarations and I/O statements used are standard Speedbase. An ACCESS statement is used to create an I/O channel to the required records. The normal Speedbase I/O verbs are then used to perform I/O. E-SMDM allows all the database access verbs, including FETCH and READ, to be used.

## 3.3 Memory Management

All memory management is automatic. 64Kb pages of memory are allocated as required to hold the tables associated with each E-SMDM I/O channel when the first I/O operation takes place, and further pages are subsequently allocated as required. Memory used by the I/O channel is automatically released when the program containing the I/O channel terminates. Since there is no physical database, there isn't anything to open. Hence there is no need to call B$OPN prior to performing I/O operations.

If necessary, the pages associated with an I/O channel can be explicitly cleared using the following call:

```
CALL B$SMDE USING $rt
```

where $*rt* is the name of the E-SMDM I/O channel. This call is functionally similar to deleting all records held within the I/O channel, but is much more memory efficient (as the actual memory pages are de-allocated). When loading a new record set into an I/O channel, any previous records should therefore be cleared using this call.

## 3.4 Window Manager

Speedbase Window manager has been enhanced to work with E-SMDM databases, and all normal operations are supported.

## 3.5 Locking

In order to maintain consistency, the standard Speedbase locking facilities have been implemented in E-SMDM even though the process is inherently single user. A record must be locked before it is rewritten. However, a lock request can obviously never fail.

## 3.6 Special considerations

Each E-SMDM I/O channel operates in its own right. This means that if a duplicate I/O channel is ACCESS'ed, then each channel will operate independently (i.e. each channel will have it's own memory pages associated with it, in which a separate record set is stored).

## 3.7 Programming Notes

Full Window Manager functionality is available with E-SMDM including full look-up capability.

However, E-SMDM requires more memory and is slightly slower than R-SMDM. Consequently, E-SMDM should only be used when its greater functionality is required.